G 3

```
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGGGGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN   GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN   GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN   GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN   GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN   GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN   GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN   GGG   GGGGGGGGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN   GGG   GGGGGGGGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN   GGG   GGGGGGGGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG         GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG         GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN   GGG         GGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCCC  HHH         HHH  NNN         NNN      GGGGGGGGG
```

```
**FILE**ID**EXCPDP                                   M  7




EEEEEEEEEE  XX      XX    CCCCCCCC  PPPPPPPP    DDDDDDDD    PPPPPPPP
EEEEEEEEEE  XX      XX    CCCCCCCC  PPPPPPPP    DDDDDDDD    PPPPPPPP
EE          XX      XX  CC          PP      PP  DD      DD  PP      PP
EE          XX      XX  CC          PP      PP  DD      DD  PP      PP
EE            XX  XX    CC          PP      PP  DD      DD  PP      PP
EE            XX  XX    CC          PP      PP  DD      DD  PP      PP
EEEEEEEE        XX      CC          PPPPPPPP    DD      DD  PPPPPPPP
EEEEEEEE        XX      CC          PPPPPPPP    DD      DD  PPPPPPPP
EE            XX  XX    CC          PP          DD      DD  PP
EE            XX  XX    CC          PP          DD      DD  PP
EE          XX      XX  CC          PP          DD      DD  PP              ....
EE          XX      XX  CC          PP          DD      DD  PP              ....
EEEEEEEEEE  XX      XX    CCCCCCCC  PP          DDDDDDDD    PP              ....
EEEEEEEEEE  XX      XX    CCCCCCCC  PP          DDDDDDDD    PP              ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE  exch$pdp                                        %TITLE 'Small PDP-11 record structure routines'
0               (
0               IDENT = 'V04-000',
0               ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE, NONEXTERNAL=WORD_RELATIVE)
0               ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 ! FACILITY:     EXCHANGE - Foreign volume interchange facility
1 !
1 ! ABSTRACT:     Specific routines for record structures from small PDP-11
1 !               systems, RT-11 and DOS-11
1 !
1 ! ENVIRONMENT:  VAX/VMS User mode
1 !
1 ! AUTHOR:       CW Hobbs                CREATION DATE: 26-Aug-1982
1 !                                       Split from RT11 module: 28-Nov-1982
1 !
1 ! MODIFIED BY:
1 !
1 !
1 !--
1
1 ! Include files:
1 !
1 MACRO $module_name_string = 'exch$pdp' %;        ! The require file needs to know our module name
1 REQUIRE 'SRC$:EXCREQ'                           ! Facility-wide require file
1     ;
```

```
%SBTTL 'Module table of contents'

! Module table of contents:
!
FORWARD ROUTINE
        pdp_buffer_advance_read,                        ! Read some more data into the ctx buffer
        pdp_buffer_advance_write,                       ! Write some data from the ctx buffer
        pdp_buffer_check  : jsb_r2r3,                   ! Check the buffer
        pdp_buffer_update : jsb_r2r3,                   ! Update the buffer pointers in the context block
        pdp_check_ctx            : NOVALUE,             ! Check the context block for consistency
        pdp_copy_binary_record : NOVALUE,               ! Copy a formatted-binary record
        pdp_copy_stream_record,                         ! Copy a record to a stream format record
    exch$pdp_filter_filename,                           ! Remove invalid characters from a filename
        pdp_find_binary_record,                         ! Find a formatted binary record in a given buffer
        pdp_find_stream_record,                         ! Find a stream record in a given buffer
    exch$pdp_flush_write_buffer,                        ! Flush any records waiting in the output buffer
    exch$pdp_get,                                       ! Get routine dispatch
        pdp_get_binary : jsb_get,                       ! Get formatted binary record
        pdp_get_fixed  : jsb_get,                       ! Get fixed-length record
        pdp_get_stream : jsb_get,                       ! Get stream format record
    exch$pdp_put,                                       ! Put dispatcher
        pdp_put_binary : jsb_put,                       ! Put formatted binary record
        pdp_put_fixed  : jsb_put,                       ! Put fixed-length record
        pdp_put_stream : jsb_put                        ! Put stream format record
    ;

! EXCHANGE facility routines
!
EXTERNAL ROUTINE
    exch$io_dos11_read,                                 ! Read blocks from a sequential device
    exch$io_dos11_skip_record,                          ! Space over blocks on a sequential device
    exch$io_dos11_write,                                ! Write blocks to a sequential device
    exch$io_rt11_read,                                  ! Read blocks from a random access device
    exch$io_rt11_write,                                 ! Write blocks to a random access device
    exch$rt11_bad_file  : NOVALUE,                      ! Erase an RT11 file because of error
    exch$util_vm_allocate                               ! Get some virtual memory
    ;

! Equated symbols:
!
!LITERAL
!    ;

! Bound declarations:
!
!BIND
!    ;

! Local macros
!
MACRO
    $$show_context =    $trace_print_fao ('cur !SL, byt !SL, eof !SL, base !SL, high !SL, wr !SL',
                        .ctx [ctx$l_cur_block], .ctx [ctx$l_cur_byte], .ctx [ctx$l_eof_block],
                        .ctx [ctx$l_buf_base_block], .ctx [ctx$l_buf_high_block], .ctx [ctx$l_high_block_wri
                        %;
```

```
1 GLOBAL ROUTINE pdp_buffer_advance_read (ctx : $ref_bblock) =     %SBTTL 'pdp_buffer_advance_read (ctx)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Move the current block to the leftmost position in the buffer, and read in new blocks
2 !
2 ! INPUTS:
2 !
2 !       ctx - ctx pointer to context for an open RT11 file
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_buffer_advance_read> ');
2
2 LOCAL
2     blks_in_use,
2     blks_to_read,
2     buf_start,                          ! Pointer to next byte in the buffer
2     buf_end,                            ! -> one past the end of buffer
2     buf_len,                            ! Length of good part of buffer
2     status
2     ;
2
2 BIND
2     base = ctx [ctx$l_buf_base_block],
2     buf  = ctx [ctx$a_buffer],
2     byt  = ctx [ctx$l_cur_byte],
2     cur  = ctx [ctx$l_cur_block],
2     eof  = ctx [ctx$l_eof_block],
2     high = ctx [ctx$l_buf_high_block],
2     filb = ctx [ctx$a_assoc_filb]       : $ref_bblock,
2     volb = ctx [ctx$a_assoc_volb]       : $ref_bblock
2     ;
2
2
2 $trace_print_lit ('entry');
```

```
  2
  2 $check_call (2, pdp_check_ctx, .ctx, 441);              ! $block_check (2, .ctx, (dos11ctx or rt11ctx), 441)
  2
  2 ! If the current block is at the beginning or the high block is EOF, we have made a grievous error
  2 !
  2 $$show_context;
  2 $logic_check (3, ((.cur GEQU .base) AND ((.cur LEQU .high+1) OR (.high EQL .base-1))), 214);
  2
  2 ! Get a pointer to the place to start shuffling, and a pointer to the first byte past the end of the buffer
  2 !
  2 $logic_check (2, (.buf NEQ 0), 181);
  2 buf_start = .buf + ((.cur - .base) * 512);
  2 buf_end   = .buf + ((1 + .high - .base) * 512);
  2 buf_len   = .buf_end - .buf_start;
  2 $logic_check (2, (.buf_len [SSU 65536), 116);   ! Short-sighted architects
  2
  2 ! If current block is the base block, do some more looking.
  2 !
  3 IF   (.cur EQL .base AND .high NEQ .base-1)     ! initial condition
  2 THEN
  3     BEGIN
P 3     $trace_print_fao ('*cur is base* buf_start !XL, buf !XL, buf_len !XW, ctx$k_buffer_length !XW',
  3             .buf_start, .buf, .buf_len, ctx$k_buffer_length);
  3     $logic_check (3, ((.buf_start EQL .buf) AND (.buf_len EQL ctx$k_buffer_length)), 215);
  3
  3     ! If there are non-null characters in the end of the buffer, then the record is too big and we have an e
  3     !
  3     IF CH$NEQ (0, .cur, .buf_len - .byt, .buf + .byt, 0)
  3     THEN
  4         $exch_signal_return (exch$_rectoobig, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name])
  4
  4     ! OK, we have some data in the first block, and nulls to the end of the buffer.  We will slide over the
  4     ! and refresh the end of the buffer, since stream and binary formats skip nulls.  This is done so that w
  4     ! handle a stream file with a large number of zeroed blocks at the end.
  4     !
  3     ELSE
  4         BEGIN
  4         $trace_print_lit ('*slide one block*');
  4         cur = .high;                            ! "Slide" it to the end
  4         buf_len = 512;                          ! One good block
  3         END;
  3     END
  3
  2 ELSE
  3     BEGIN
  3
  3     ! Current not the base, move the good data to the start of the buffer
  3     !
  3     $trace_print_fao ('*cur not base* buf_start !XL, buf !XL, buf_len !XW', .buf_start, .buf, .buf_len);
  3     IF .buf_start NEQ .buf
  3     THEN
  4         BEGIN
  4         $trace_print_lit ('shuffling data to the start of the buffer');
  4         CH$MOVE (.buf_len, .buf_start, .buf);
  3         END;
  2     END;
  2
```

```
;  223       0317  2  ! Change the base pointer to show what we just did, buf_high_block is still valid
;  224       0318  2  !
;  225       0319  2  base = .cur;
;  226       0320  2
;  227       0321  2  ! Read a chunk into the buffer
;  228       0322  2  !
;  229       0323  2  blks_in_use = .buf_len / 512;
;  230       0324  2  blks_to_read = ctx$k_buffer_blocks - .blks_in_use;                ! Blocks left in buffer
;  231       0325  2  IF (.eof - .high) GTR 0                                            ! Blocks left in file
;  232       0326  2  THEN
;  233       0327  2      blks_to_read = MINU (.blks_to_read, (.eof - .high));
;  234       0328  2
;  235       0329  2  ! If all of the blocks are in use, then we have no room to fit more data into the buffer.  Return with a rec
;  236       0330  2  ! error, which our caller can examine.
;  237       0331  2  !
;  238       0332  2  IF .blks_in_use GEQU ctx$k_buffer_blocks
;  239       0333  2  THEN
;  240       0334  2      RETURN exch$_stmrecfmt;
;  241       0335  2
;  242     P 0336  2  $trace_print_fao ('blocks in use !UL, blocks to read !UL, ctx$k_buffer_blocks !UL',
;  243       0337  2                  .blks_in_use, .blks_to_read, ctx$k_buffer_blocks);
;  244       0338  2  $$show_context;
;  245       0339  2  $logic_check (2, (.blks_to_read GTRU 0), 118);
;  246       0340  2
;  247       0341  2  ! Perform the appropriate read operation depending on the volume type
;  248       0342  2  !
;  249       0343  2  IF .volb [volb$b_vol_format] EQL volb$k_vfmt_rt11
;  250       0344  2  THEN
;  251       0345  3      BEGIN
;  252       0346  4      IF NOT (status = exch$io_rt11_read (.volb,                    ! All the rms stuff hangs from here
;  253       0347  4                                          .high + 1,                ! First block to read
;  254       0348  4                                          .blks_to_read,            ! Number of blocks
;  255       0349  4                                          .buf + .buf_len))         ! Address of the I/O buffer
;  256       0350  3      THEN
;  257       0351  3          RETURN .status;
;  258       0352  3      END
;  259       0353  2  ELSE
;  260       0354  3      BEGIN
;  261       0355  3      LOCAL
;  262       0356  3          bp,                                           ! Buffer pointer
;  263       0357  3          bc;                                           ! Block count
;  264       0358  3
;  265       0359  3      bc = .blks_to_read;                               ! Number of blocks to read
;  266       0360  3      bp = .buf + .buf_len;                             ! Address to put first block
;  267       0361  3
;  268       0362  3      WHILE 1
;  269       0363  3      DO
;  270       0364  4          BEGIN
;  271       0365  4
;  272       0366  4          ! Read from the tape
;  273       0367  4          !
;  274       0368  4          status = exch$io_dos11_read (   .volb,  ! All the stuff hangs from here
;  275       0369  4                                          .bp);   ! Address of the I/O buffer
;  276       0370  4
;  277       0371  4          ! If the read didn't work, do some checking
;  278       0372  4          !
;  279       0373  4          IF NOT .status
;   280          0374  4              THEN
;   281          0375  5                  BEGIN
;   282          0376  5                  IF .status EQL ss$_endoffile
;   283          0377  5                     OR
;   284          0378  5                      .status EQL ss$_endoftape
;   285          0379  5                  THEN
;   286          0380  6                      BEGIN
;   287          0381  6                      $trace_print_lit ('registered END-OF-FILE');
;   288          0382  6                      $$show_context;
;   289          0383  6                      eof = MAX (0, (.high + (.blks_to_read - .bc))); ! Set the eof block to zero or more
;   290          0384  6                      blks_to_read = .blks_to_read - .bc;               ! Adjust so that high block gets set right;
;   291          0385  6                      EXITLOOP;
;   292          0386  6                      END
;   293          0387  5                  ELSE
;   294          0388  5                      RETURN .status;                           ! Return the error status
;   295          0389  4                  END;
;   296          0390  4
;   297          0391  4              ! Adjust our pointers
;   298          0392  4              !
;   299          0393  4              bp = .bp + 512;                           ! Move to the next block
;   300          0394  4              bc = .bc - 1;
;   301          0395  4              IF .bc LEQ 0 THEN EXITLOOP;               ! Exit if all have been read
;   302          0396  4
;   303          0397  3              END;
;   304          0398  2          END;
;   305          0399  2
;   306          0400  2      ! Change the high block pointer to show what we just did
;   307          0401  2      !
;   308          0402  2      high = .high + .blks_to_read;
;   309          0403  2
;   310          0404  2      RETURN true;
;   311          0405  2
;   312          0406  1      END;


                                                                  .TITLE  EXCH$PDP Small PDP-11 record structure routines
                                                                  .IDENT  \V04-000\

                                                                  .EXTRN  EXCH$IO_DOS11_READ
                                                                  .EXTRN  EXCH$IO_DOS11_SKIP_RECORD
                                                                  .EXTRN  EXCH$IO_DOS11_WRITE
                                                                  .EXTRN  EXCH$IO_RT11_READ
                                                                  .EXTRN  EXCH$IO_RT11_WRITE
                                                                  .EXTRN  EXCH$RT11_BAD_FILE
                                                                  .EXTRN  EXCH$UTIL_VM_ALLOCATE
                                                                  .EXTRN  PDP_CHECK_CTX, EXCH$_BADLOGIC
                                                                  .EXTRN  EXCH$_RECTOOBIG
                                                                  .EXTRN  EXCH$_STMRECFMT

                                                                  .PSECT  EXCH$PDP_CODE,NOWRT,2

                                            0FFC 00000            .ENTRY  PDP_BUFFER_ADVANCE_READ, Save R2,R3,R4,R5,-  ; 0203
                                                                          R6,R7,R8,R9,R10,R11                           ;
                           5B 00000000G   00  9E 00002            MOVAB   LIB$STOP, R11                                 ;
                           5A 00000000G   8F  D0 00009            MOVL    #EXCH$_BADLOGIC, R10                          ;
                           57         04  AC  D0 00010            MOVL    CTX, R7                                       ; 0248
                           7E       01B9  8F  3C 00014            MOVZWL  #441, -(SP)                                   ; 0261
```

```
                                              57  DD 00019          PUSHL    R7                                               :
                      00000000G   00          02  FB 0001B          CALLS    #2, PDP_CHECK_CTX                                :
                                  58     18   A7  D0 00022          MOVL     24(R7), R8                                       : 0270
                                              0B  12 00026          BNEQ     1$                                               :
                                  7E     B5   8F  9A 00028          MOVZBL   #181, -(SP)                                      :
                                              01  DD 0002C          PUSHL    #1                                               :
                                              5A  DD 0002E          PUSHL    R10                                              :
                                  6B          03  FB 00030          CALLS    #3, LIB$STOP                                     :
                  53      1C      A7     2C   A7  C3 00033 1$:      SUBL3    44(R7), 28(R7), R3                               : 0271
                  53              53          09  78 00039          ASHL     #9, R3, R3                                       :
                  52              53          58  C1 0003D          ADDL3    R8, R3, BUF_START                                :
                                  59     30   A7  D0 00041          MOVL     48(R7), R9                                       : 0272
                  50              59     2C   A7  C3 00045          SUBL3    44(R7), R9, R0                                   :
                  50              50          09  78 0004A          ASHL     #9, R0, R0                                       :
                                  50   0200 C048  9E 0004E          MOVAB    512(R0)[R8], BUF_END                             :
                  56              50          52  C3 00054          SUBL3    BUF_START, BUF_END, BUF_LEN                      : 0273
                      00010000    8F          56  D1 00058          CMPL     BUF_LEN, #65536                                  : 0274
                                              0B  1F 0005F          BLSSU    2$                                               :
                                  7E     74   8F  9A 00061          MOVZBL   #116, -(SP)                                      :
                                              01  DD 00065          PUSHL    #1                                               :
                                              5A  DD 00067          PUSHL    R10                                              :
                                  6B          03  FB 00069          CALLS    #3, LIB$STOP                                     :
                          2C      A7     1C   A7  D1 0006C 2$:      CMPL     28(R7), 44(R7)                                   : 0278
                                              45  12 00071          BNEQ     4$                                               :
                  50      2C      A7          01  C3 00073          SUBL3    #1, 44(R7), R0                                   :
                                  50          59  D1 00078          CMPL     R9, R0                                           :
                                              3B  13 0007B          BEQL     4$                                               :
                  50              56     24   A7  C3 0007D          SUBL3    36(R7), BUF_LEN, R0                              : 0287
     50           00      1C      B7          00  2D 00082          CMPC5    #0, @28(R7), #0, R0, @36(R7)[R8]                 :
                                       24 B748       00088                                                                    :
                                              20  13 0008B          BEQL     3$                                               :
                                  52 00000000G 8F D0 0008D          MOVL     #EXCH$_RECTOOBIG, TEMP                           : 0289
                                  50     10   A7  D0 00094          MOVL     16(R7), R0                                       :
                                         5A   A0  9F 00098          PUSHAB   90(R0)                                           :
                                         3A   A0  DD 0009B          PUSHL    58(R0)                                           :
                                              02  DD 0009E          PUSHL    #2                                               :
                                              52  DD 000A0          PUSHL    TEMP                                             :
                      00000000G   00          04  FB 000A2          CALLS    #4, LIB$SIGNAL                                   :
                                  50          52  D0 000A9          MOVL     TEMP, R0                                         :
                                                  04 000AC          RET                                                       :
                          1C      A7          59  D0 000AD 3$:      MOVL     R9, 28(R7)                                       : 0298
                                  56   0200   8F  3C 000B1          MOVZWL   #512, BUF_LEN                                    : 0299
                                              09  11 000B6          BRB      5$                                               : 0278
                                  58          52  D1 000B8 4$:      CMPL     BUF_START, R8                                    : 0309
                                              04  13 000BB          BEQL     5$                                               :
                  68              62          56  28 000BD          MOVC3    BUF_LEN, (BUF_START), (R8)                       : 0313
                          2C      A7     1C   A7  D0 000C1 5$:      MOVL     28(R7), 44(R7)                                   : 0319
                  50              56 00000200 8F  C7 000C6          DIVL3    #512, BUF_LEN, BLKS_IN_USE                       : 0323
                  52              0C          50  C3 000CE          SUBL3    BLKS_IN_USE, #12, BLKS_TO_READ                   : 0324
                                  59     20   A7  D1 000D2          CMPL     32(R7), R9                                       : 0325
                                              13  15 000D6          BLEQ     7$                                               :
                  53      20      A7          59  C3 000D8          SUBL3    R9, 32(R7), R3                                   : 0327
                                  51          52  D0 000DD          MOVL     BLKS_TO_READ, R1                                 :
                                  53          51  D1 000E0          CMPL     R1, R3                                           :
                                              03  1B 000E3          BLEQU    6$                                               :
                                  51          53  D0 000E5          MOVL     R3, R1                                           :
                                  52          51  D0 000E8 6$:      MOVL     R1, BLKS_TO_READ                                 :
```

```
                        0C            50 D1 000EB 7$:     CMPL    BLKS_IN_USE, #12                          ; 0332
                                      08 1F 000EE         BLSSU   8$                                        ;
                        50 00000000G  8F D0 000F0         MOVL    #EXCH$_STMRECFMT, R0                      ; 0334
                                      04 000F7            RET                                               ;
                                      52 D5 000F8 8$:     TSTL    BLKS_TO_READ                              ; 0339
                                      0B 12 000FA         BNEQ    9$                                        ;
                        7E       76   8F 9A 000FC         MOVZBL  #118, -(SP)                               ;
                                      01 DD 00100         PUSHL   #1                                        ;
                                      5A DD 00102         PUSHL   R10                                       ;
                        6B            03 FB 00104         CALLS   #3, LIB$STOP                              ;
                        53       14   A7 D0 00107 9$:     MOVL    20(R7), R3                                ; 0343
                        03       58   A3 91 0010B         CMPB    88(R3), #3                                ;
                                      15 12 0010F         BNEQ    10$                                       ;
                               6648   9F 00111            PUSHAB  (BUF_LEN)[R8]                             ; 0349
                                      52 DD 00114         PUSHL   BLKS_TO_READ                              ; 0348
                                 01   A9 9F 00116         PUSHAB  1(R9)                                     ; 0347
                                      53 DD 00119         PUSHL   R3                                        ; 0346
              00000000G EF            04 FB 0011B         CALLS   #4, EXCH$IO_RT11_READ                     ;
                        42            50 E8 00122         BLBS    STATUS, 15$                               ;
                                      04 00125            RET                                               ; 0351
                        54            52 D0 00126 10$:    MOVL    BLKS_TO_READ, BC                          ; 0359
         55             58            56 C1 00129         ADDL3   BUF_LEN, R8, BP                           ; 0360
                                      28 BB 0012D 11$:    PUSHR   #^M<R3,R5>                                ; 0368
              00000000G EF            02 FB 0012F         CALLS   #2, EXCH$IO_DOS11_READ                    ;
                        26            50 E8 00136         BLBS    STATUS, 14$                               ; 0373
              00000870  8F            50 D1 00139         CMPL    STATUS, #2160                             ; 0376
                        09            13 00140            BEQL    12$                                       ;
              00000878  8F            50 D1 00142         CMPL    STATUS, #2168                             ; 0378
                        23            12 00149            BNEQ    16$                                       ;
         51             52            54 C3 0014B 12$:    SUBL3   BC, BLKS_TO_READ, R1                      ; 0383
                        51            59 C0 0014F         ADDL2   R9, R1                                    ;
                        02            18 00152            BGEQ    13$                                       ;
                                      51 D4 00154         CLRL    R1                                        ;
                   20   A7            51 D0 00156 13$:    MOVL    R1, 32(R7)                                ;
                        52            54 C2 0015A         SUBL2   BC, BLKS_TO_READ                          ; 0384
                                      08 11 0015D         BRB     15$                                       ; 0380
                        55     0200   C5 9E 0015F 14$:    MOVAB   512(R5), BP                               ; 0393
                        C6            54 F5 00164         SOBGTR  BC, 11$                                   ; 0394
                   30   A7            52 C0 00167 15$:    ADDL2   BLKS_TO_READ, 48(R7)                      ; 0402
                        50            01 D0 0016B         MOVL    #1, R0                                    ; 0404
                                      04 0016E 16$:       RET                                               ; 0406

; Routine Size:  367 bytes,    Routine Base:  EXCH$PDP_CODE + 0000
```

```
1 GLOBAL ROUTINE pdp_buffer_advance_write (ctx : $ref_bblock) =    %SBTTL 'pdp_buffer_advance_write (ctx)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Write the complete blocks in the buffer, then move the current block to the leftmost position in the
2 !       buffer.
2 !
2 ! INPUTS:
2 !
2 !       ctx - ctx pointer to context for an open RT11 file
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_buffer_advance_write> ');
2
2 LOCAL
2     temp,
2     blks_to_write,
2     buf_start,                            ! Pointer to next byte in the buffer
2     buf_end,                              ! -> one past the end of buffer
2     buf_len,                              ! Length of good part of buffer
2     status
2     ;
2
2 BIND
2     base = ctx [ctx$l_buf_base_block],
2     buf  = ctx [ctx$a_buffer],
2     cur  = ctx [ctx$l_cur_block],
2     eof  = ctx [ctx$l_eof_block],
2     high = ctx [ctx$l_buf_high_block],
2     filb = ctx [ctx$a_assoc_filb]         : $ref_bblock,
2     volb = ctx [ctx$a_assoc_volb]         : $ref_bblock
2     ;
2
2
2 $trace_print_lit ('entry');
```

```
  2
  2 $check_call (2, pdp_check_ctx, .ctx, 458);        ! $block_check (2, .ctx, (), 458);
  2
  2 ! If the current block is before the beginning or the high block past EOF, we have made a grievous error
  2 !
  2 $logic_check (2, ((.cur GEQU .base) AND (.high LEQU .eof)), 242);
  2
  2 ! How many full blocks do we have?
  2 !
  2 blks_to_write = .cur - .base;
  2
  2 ! Get a pointer to the first partial block, the end of the buffer, and the length from the first partial to
  2 ! the end of the block
  2 !
  2 $logic_check (2, (.buf NEQ 0), 194);
  2 buf_start = .buf + ((.cur - .base) * 512);
  2 buf_end   = .buf + ((1 + .high - .base) * 512);
  2 buf_len   = .buf_end - .buf_start;
  2 $logic_check (2, (.buf_len LSSU 65536), 173);              ! Short-sighted architects
  2
  2 ! Do a flush operation if necessary.  The final partial block will be padded with nulls.
  2 !
  4 IF ((.ctx [ctx$v_flush])                            ! Has a flush been requested
  3   AND
  3     (.ctx [ctx$l_cur_byte] NEQ 0))                   ! And is there a partial block waiting
  2 THEN
  3     BEGIN
  3
  3     blks_to_write = .blks_to_write + 1;             ! Adjust the block count for the partial
  3
  3     CH$FILL (0, .buf_len - .ctx [ctx$l_cur_byte], .buf_start + .ctx [ctx$l_cur_byte]);
  2     END;
  2
  2 ! If we are flushing, set the eof block so that we may update the entry when we close (DOS-11 only)
  2 !
  3 IF (.ctx [ctx$v_flush])
  2   AND
  3    (.eof EQL -1)                                    ! DOS-11 has -1 for an EOF block
  2 THEN
  3     BEGIN
  3     $trace_print_lit ('flushing...');
  3     $$show_context;
  3     eof = .base + .blks_to_write - 1;
  2     END;
  2
P 2 $trace_print_fao ('buf !XL, buf_start !XL, buf_end !XL, buf_len !XL, blocks to write !UL',
  2                     .buf, .buf_start, .buf_end, .buf_len, .blks_to_write);
  2 $$show_context;
  2
  2 ! If no blocks, we don't have any more to do
  2 !
  2 IF .blks_to_write EQL 0
  2 THEN
  2     RETURN true;
  2
  2 ! Write the front chunk from the buffer, operation depends on the volume type
  2 !
```

```
2 $logic_check (2, ((.blks_to_write GTRU 0) AND (.blks_to_write LEQU ctx$k_buffer_blocks)), 174);
2 IF .volb [volb$b_vol_format] EQL volb$k_vfmt_rt11
2 THEN
3     BEGIN
3
4     IF NOT (status = exch$io_rt11_write (    .volb,                    ! All the rms stuff hangs from here
4                                              .base,                    ! First block to write
4                                              .blks_to_write,           ! Number of blocks
4                                              .buf))                    ! Address of the I/O buffer
3     THEN
4         BEGIN
4         exch$rt11_bad_file (.filb);
4         RETURN .status;
3         END;
3     END
2 ELSE
3     BEGIN
3     LOCAL
3         bl,                                        ! Buffer length
3         bp,                                        ! Buffer pointer
3         bc;                                        ! Block count
3
3     bl = 512;                                      ! Most blocks are 512 bytes
3     bc = .blks_to_write;                           ! Number of blocks to write
3     bp = .buf;                                     ! Address to find first block
3
3     WHILE 1
3     DO
4         BEGIN
4
4         ! See if we are writing a final, short block
4         !
4         IF .ctx [ctx$v_flush]                      ! Only if we are flushing
4         THEN
4             IF .bc EQL 1                           ! And if we are writing the last block
4             THEN
4                 IF .ctx [ctx$l_cur_byte] NEQ 0     ! And if the block is partial
4                 THEN
4                     bl = .ctx [ctx$l_cur_byte];    ! Then the length is that partial
4
4         ! Write to the tape
4         !
4         status = exch$io_dos11_write (   .volb,    ! All the stuff hangs from here
4                                          .bp,      ! Address of the I/O buffer
4                                          .bl);     ! Length of the I/O buffer
4
4         ! If the write didn't work, mark the buffer as empty before returning
4         !
4         IF NOT .status                                         ! Probably ss$_endoftape
4         THEN
5             BEGIN
5             cur = .base + (.blks_to_write - .bc);              ! Set cur to high block written before error
5             base = .cur;                                       ! Say that base is the current
5             ctx [ctx$l_cur_byte] = 0;                          ! Say that no bytes in last block
5             exch$io_dos11_skip_record (.volb, -1);             ! Backup one record
5             RETURN .status;                                    ! Return the error status
4             END;
```

```
;   485          0578 4
;   486          0579 4          ! Adjust our pointers
;   487          0580 4          !
;   488          0581 4          bp = .bp + 512;                          ! Move to the next block
;   489          0582 4          bc = .bc - 1;
;   490          0583 4          IF .bc LEQ 0 THEN EXITLOOP;               ! Exit if all have been read
;   491          0584 4
;   492          0585 3          END;
;   493          0586 2      END;
;   494          0587 2
;   495          0588 2  ! If we have exceeded the previous high water mark, save the new mark
;   496          0589 2  !
;   497          0590 2  temp = (.base + (.blks_to_write-1));
;   498          0591 2  IF .temp GTRU .ctx [ctx$l_high_block_written]
;   499          0592 2  THEN
;   500          0593 2      ctx [ctx$l_high_block_written] = .temp;
;   501          0594 2
;   502          0595 2  ! Move the good data to the start of the buffer
;   503          0596 2  !
;   504          0597 2  CH$MOVE (.buf_len, .buf_start, .buf);
;   505          0598 2
;   506          0599 2  ! Change the base pointer to show what we just did, buf_high_block is still valid
;   507          0600 2  !
;   508          0601 2  base = .cur;
;   509          0602 2
;   510          0603 2  ! Change the high block pointer to show what we just did
;   511          0604 2  !
;   512          0605 2  high = MINU ((.high + .blks_to_write), .eof);
;   513          0606 2
;   514          0607 2  $trace_print_lit ('context at exit');
;   515          0608 2  $$show_context;
;   516          0609 2
;   517          0610 2  RETURN true;
;   518          0611 2
;   519          0612 1  END;



                                         0FFC 00000          .ENTRY  PDP_BUFFER_ADVANCE_WRITE, Save R2,R3,R4,R5,-; 0407
                                                                     R6,R7,R8,R9,R10,R11                          :
                    5E              04    C2 00002          SUBL2   #4, SP                                       :
                    58        04    AC    D0 00005          MOVL    CTX, R8                                      ; 0453
                    5A        2C    A8    9E 00009          MOVAB   44(R8), R10                                  :
                    7E      01CA    8F    3C 0000D          MOVZWL  #458, -(SP)                                  ; 0465
                                    58    DD 00012          PUSHL   R8                                           :
           00000000G  00            02    FB 00014          CALLS   #2, PDP_CHECK_CTX                            :
                    6A        1C    A8    D1 0001B          CMPL    28(R8), (R10)                                ; 0469
                                    07    1F 0001F          BLSSU   1$                                           :
                20  A8        30    A8    D1 00021          CMPL    48(R8), 32(R8)                               :
                                    13    1B 00026          BLEQU   2$                                           :
                    7E        F2    8F    9A 00028 1$:      MOVZBL  #242, -(SP)                                  :
                                    01    DD 0002C          PUSHL   #1                                           :
                         00000000G  8F    DD 0002E          PUSHL   #EXCH$_BADLOGIC                              :
           00000000G  00            03    FB 00034          CALLS   #3, LIB$STOP                                 :
        52        1C  A8            6A    C3 0003B 2$:      SUBL3   (R10), 28(R8), R2                            ; 0473
```

```
                                                      M  8
EXCH$PDP       Small PDP-11 record structure routines       16-Sep-1984 01:11:46    VAX-11 Bliss-32 V4.0-742          Page  13
V04-000        pdp_buffer_advance_write (ctx)               14-Sep-1984 12:29:07    [EXCHNG.SRC]EXCPDP.B32;1                 (4)

                         56                 52  D0 00040          MOVL    R2, BLKS_TO_WRITE                  :
                         59         18      A8  D0 00043          MOVL    24(R8), R9                         : 0478
                                            13  12 00047          BNEQ    3$                                 :
                         7E         C2      8F  9A 00049          MOVZBL  #194, -(SP)                        :
                                            01  DD 0004D          PUSHL   #1                                 :
                              00000000G     8F  DD 0004F          PUSHL   #EXCH$_BADLOGIC                    :
               00000000G 00                 03  FB 00055          CALLS   #3, LIB$STOP                       :
            50           52                 09  78 0005C 3$:      ASHL    #9, R2, R0                         : 0479
            5B           50                 59  C1 00060          ADDL3   R9, R0, BUF_START                  :
            50        30 A8                 6A  C3 00064          SUBL3   (R10), 48(R8), R0                  : 0480
            50           50                 09  78 00069          ASHL    #9, R0, R0                         :
                         50       0200    C049  9E 0006D          MOVAB   512(R0)[R9], BUF_END               :
            57           50                 5B  C3 00073          SUBL3   BUF_START, BUF_END, BUF_LEN        : 0481
               00010000  8F                 57  D1 00077          CMPL    BUF_LEN, #65536                    : 0482
                                            13  1F 0007E          BLSSU   4$                                 :
                         7E         AD      8F  9A 00080          MOVZBL  #173, -(SP)                        :
                                            01  DD 00084          PUSHL   #1                                 :
                              00000000G     8F  DD 00086          PUSHL   #EXCH$_BADLOGIC                    :
               00000000G 00                 03  FB 0008C          CALLS   #3, LIB$STOP                       :
            2C        28 A8                 02  E1 00093 4$:      BBC     #2, 40(R8), 6$                     : 0486
                                    24      A8  D5 00098          TSTL    36(R8)                             : 0488
                                            0F  13 0009B          BEQL    5$                                 :
                                            56  D6 0009D          INCL    BLKS_TO_WRITE                      : 0492
            50           57         24      A8  C3 0009F          SUBL3   36(R8), BUF_LEN, R0                : 0494
    50      00           6E                 00  2C 000A4          MOVC5   #0, (SP), #0, R0, @36(R8)[BUF_START]  : 0494
                                    24    B84B     000A9                                                     :
            13        28 A8                 02  E1 000AC 5$:      BBC     #2, 40(R8), 6$                     : 0499
               FFFFFFFF  8F         20      A8  D1 000B1          CMPL    32(R8), #-1                        : 0501
                                            09  12 000B9          BNEQ    6$                                 :
            50           6A                 56  C1 000BB          ADDL3   BLKS_TO_WRITE, (R10), R0           : 0506
                      20 A8         FF      A0  9E 000BF          MOVAB   -1(R0), 32(R8)                     :
                                            56  D5 000C4 6$:      TSTL    BLKS_TO_WRITE                      : 0515
                                            03  12 000C6          BNEQ    7$                                 :
                                          00C5  31 000C8          BRW     17$                                :
                         0C                 56  D1 000CB 7$:      CMPL    BLKS_TO_WRITE, #12                 : 0521
                                            13  1B 000CE          BLEQU   8$                                 :
                         7E         AE      8F  9A 000D0          MOVZBL  #174, -(SP)                        :
                                            01  DD 000D4          PUSHL   #1                                 :
                              00000000G     8F  DD 000D6          PUSHL   #EXCH$_BADLOGIC                    :
               00000000G 00                 03  FB 000DC          CALLS   #3, LIB$STOP                       :
                         52         14      A8  D0 000E3 8$:      MOVL    20(R8), R2                         : 0522
                         03         58      A2  91 000E7          CMPB    88(R2), #3                         :
                                            21  12 000EB          BNEQ    9$                                 :
                                  0240      8F  BB 000ED          PUSHR   #^M<R6,R9>                         : 0528
                                            6A  DD 000F1          PUSHL   (R10)                              : 0527
                                            52  DD 000F3          PUSHL   R2                                 : 0526
               00000000G EF                 04  FB 000F5          CALLS   #4, EXCH$IO_RT11_WRITE             :
                         54                 50  D0 000FC          MOVL    R0, STATUS                         :
                         63                 54  E8 000FF          BLBS    STATUS, 14$                        :
                                    10      A8  DD 00102          PUSHL   16(R8)                             : 0532
               00000000G EF                 01  FB 00105          CALLS   #1, EXCH$RT11_BAD_FILE             :
                                            4B  11 0010C          BRB     12$                                : 0533
                         6E       0200      8F  3C 0010E 9$:      MOVZWL  #512, BL                           : 0543
                         53                 56  D0 00113          MOVL    BLKS_TO_WRITE, BC                  : 0544
                         55                 59  D0 00116          MOVL    R9, BP                             : 0545
            0E        28 A8                 02  E1 00119 10$:     BBC     #2, 40(R8), 11$                    : 0553
                         01                 53  D1 0011E          CMPL    BC, #1                             : 0555
```

```
                                                          N  8
EXCH$PDP        Small PDP-11 record structure routines   16-Sep-1984 01:11:46   VAX-11 Bliss-32 V4.0-742          Page  14
V04-000         pdp_buffer_advance_write (ctx)           14-Sep-1984 12:29:07   [EXCHNG.SRC]EXCPDP.B32;1                 (4)

                                         09  12 00121         BNEQ    11$                                     ;
                                   24    A8  D5 00123         TSTL    36(R8)                                  ; 0557
                                         04  13 00126         BEQL    11$                                     ;
                         6E        24    A8  D0 00128         MOVL    36(R8), BL                              ; 0559
                                         6E  DD 0012C  11$:   PUSHL   BL                                      ; 0565
                                         24  BB 0012E         PUSHR   #^M<R2,R5>                              ; 0563
               00000000G EF              03  FB 00130         CALLS   #3, EXCH$IO_DOS11_WRITE                 ;
                         54              50  D0 00137         MOVL    R0, STATUS                              ;
                         20              54  E8 0013A         BLBS    STATUS, 13$                             ; 0569
          50             56              53  C3 0013D         SUBL3   BC, BLKS_TO_WRITE, R0                   ; 0572
    1C    A8             50              6A  C1 00141         ADDL3   (R10), R0, 28(R8)                       ;
                         6A        1C    A8  D0 00146         MOVL    28(R8), (R10)                           ; 0573
                                   24    A8  D4 0014A         CLRL    36(R8)                                  ; 0574
                         7E              01  CE 0014D         MNEGL   #1, -(SP)                               ; 0575
                                         52  DD 00150         PUSHL   R2                                      ;
               00000000G EF              02  FB 00152         CALLS   #2, EXCH$IO_DOS11_SKIP_RECORD           ;
                         50              54  D0 00159  12$:   MOVL    STATUS, R0                              ; 0576
                                             04 0015C         RET                                             ;
                         55      0200    C5  9E 0015D  13$:   MOVAB   512(R5), BP                             ; 0581
                         B4              53  F5 00162         SOBGTR  BC, 10$                                 ; 0582
          50             6A              56  C1 00165  14$:   ADDL3   BLKS_TO_WRITE, (R10), R0                ; 0590
                                         50  D7 00169         DECL    TEMP                                    ;
                   34    A8              50  D1 0016B         CMPL    TEMP, 52(R8)                            ; 0591
                                         04  1B 0016F         BLEQU   15$                                     ;
                   34    A8              50  D0 00171         MOVL    TEMP, 52(R8)                            ; 0593
          69             6B              57  28 00175  15$:   MOVC3   BUF_LEN, (BUF_START), (R9)              ; 0597
                         6A        1C    A8  D0 00179         MOVL    28(R8), (R10)                           ; 0601
          50             56        30    A8  C1 0017D         ADDL3   48(R8), BLKS_TO_WRITE, R0               ; 0605
                   20    A8              50  D1 00182         CMPL    R0, 32(R8)                              ;
                                         04  1B 00186         BLEQU   16$                                     ;
                         50        20    A8  D0 00188         MOVL    32(R8), R0                              ;
                   30    A8              50  D0 0018C  16$:   MOVL    R0, 48(R8)                              ;
                         50              01  D0 00190  17$:   MOVL    #1, R0                                  ; 0610
                                             04 00193         RET                                             ; 0612

; Routine Size:  404 bytes,    Routine Base:  EXCH$PDP_CODE + 016F
```

```
1 GLOBAL ROUTINE pdp_buffer_check (ctx : $ref_bblock, out_filb : $ref_bblock) : jsb_r2r3 =        %SBTTL 'pdp_
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Handle the situation of buffer overflow by either writing some blocks or signalling EOF.
2 !
2 ! INPUTS:
2 !
2 !       ctx      - Output file context block
2 !       out_filb - Output file block
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_buffer_check> ');
2
2 REGISTER
2     tmp
2     ;
2
2 $debug_print_lit ('entry');
2
2 ! If the EOF block is in the buffer
2 !
2 IF .ctx [ctx$l_buf_high_block] GEQU .ctx [ctx$l_eof_block]
2 THEN
2
2     ! Don't have any more room at the inn
2     !
3     $exch_signal_return (exch$_rtouteof, 2, .out_filb [filb$l_result_name_len], out_filb [filb$t_result_name
3
3 ! Otherwise, write some data and recursively retry the put
3 !
2 ELSE
3     BEGIN
4     IF NOT (tmp = pdp_buffer_advance_write (.ctx))
3     THEN
```

```
;  578         0670  3            RETURN .tmp;
;  579         0671  3        RETURN exch$pdp_put ();                  ! And then try it again
;  580         0672  2        END;
;  581         0673  2
;  582         0674  1 END;



                                                                  .EXTRN  EXCH$_RTOUTEOF

                    20   A2       30   A2  D1 00000 PDP_BUFFER_CHECK::
                                                                  CMPL    48(CTX), 32(CTX)                             ; 0657
                                           1C  1F 00005           BLSSU   1$
                         52 00000000G  8F  D0 00007           MOVL    #EXCH$_RTOUTEOF, TEMP                        ; 0662
                                       5A  A3  9F 0000E           PUSHAB  90(OUT_FILB)
                                       3A  A3  DD 00011           PUSHL   58(OUT_FILB)
                                           02  DD 00014           PUSHL   #2
                                           52  DD 00016           PUSHL   TEMP
                00000000G  00              04  FB 00018           CALLS   #4, LIB$SIGNAL
                           50              52  D0 0001F           MOVL    TEMP, R0                                     ; 0667
                                           05 00022           RSB
                                           52  DD 00023 1$:       PUSHL   CTX                                          ; 0668
                      FE42  CF             01  FB 00025           CALLS   #1, PDP_BUFFER_ADVANCE_WRITE
                            05             50  E9 0002A           BLBC    TMP, 2$
                     0000V  CF             00  FB 0002D           CALLS   #0, EXCH$PDP_PUT                             ; 0671
                                           05 00032 2$:       RSB                                                  ; 0674

; Routine Size:  51 bytes,    Routine Base:  EXCH$PDP_CODE + 0303
```

```
1 GLOBAL ROUTINE pdp_buffer_update (ctx : $ref_bblock, next_buf) : jsb_r2r3 =      %SBTTL 'pdp_buffer_update'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Update the current byte information in the context
2 !
2 ! INPUTS:
2 !
2 !       ctx       - Output file context block
2 !       next_buf  - New current record pointer
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_buffer_update> ');
2
2 REGISTER
2     five12,
2     tmp
2     ;
2
2 $debug_print_lit ('entry');
2
2 ! Update the next record position
2 !
2 five12 = 512;
2 $logic_check (2, (.ctx [ctx$a_buffer] NEQ 0), 201);
2 tmp = .next_buf - .ctx [ctx$a_buffer];  ! Save the updated position for the next put
2 ctx [ctx$l_cur_byte]  = .tmp MOD .five12;
2 ctx [ctx$l_cur_block] = (.tmp / .five12) + .ctx [ctx$l_buf_base_block];
2
2 RETURN true;
2
1 END;
```

```
                                           53  DD 00000 PDP_BUFFER_UPDATE::
                                                                 PUSHL   R3                                    ; 0675
                            53     0200   8F   3C 00002          MOVZWL  #512, FIVE12                          ; 0720
                                     18   A2   D5 00007          TSTL    24(CTX)                               ; 0721
                                          13   12 0000A          BNEQ    1$                                    ;
                            7E       C9   8F   9A 0000C          MOVZBL  #201, -(SP)                           ;
                                          01   DD 00010          PUSHL   #1                                    ;
                                 00000000G 8F  DD 00012          PUSHL   #EXCH$_BADLOGIC                       ;
                      00000000G  00       03   FB 00018          CALLS   #3, LIB$STOP                          ;
                  50             6E   18  A2   C3 0001F 1$:      SUBL3   24(CTX), NEXT_BUF, TMP                ; 0722
       7E         00             50       01   7A 00024          EMUL    #1, TMP, #0, -(SP)                    ; 0723
       51         51             8E       53   7B 00029          EDIV    FIVE12, (SP)+, R1, R1                 ;
                             24  A2       51   D0 0002E          MOVL    R1, 36(CTX)                           ;
                                 50       53   C6 00032          DIVL2   FIVE12, R0                            ; 0724
                             1C  A2  2C B240   9E 00035          MOVAB   @44(CTX)[R0], 28(CTX)                 ;
                                 50       01   D0 0003B          MOVL    #1, R0                                ; 0726
                                 5E       04   C0 0003E          ADDL2   #4, SP                                ; 0728
                                               05 00041          RSB                                           ;

; Routine Size:  66 bytes,    Routine Base:  EXCH$PDP_CODE + 0336
```

```
1 GLOBAL ROUTINE pdp_check_ctx (ctx : $ref_bblock, code) : NOVALUE =      %SBTTL 'pdp_check_ctx'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Check for a valid context block
2 !
2 ! INPUTS:
2 !
2 !       ctx       - Output file context block
2 !       code      - Error code to use if the check fails
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_check_ctx> ');
2
2 LOCAL
2     size,
2     type
2     ;
2
2 BIND
2     filb = ctx [ctx$a_assoc_filb]        : $ref_bblock,
2     volb = ctx [ctx$a_assoc_volb]        : $ref_bblock
2     ;
2
2 $debug_print_lit ('entry');
2
2 ! The context block must exist
2 !
2 IF .ctx EQL 0
2 THEN
2     $exch_signal_stop (exch$_blockcheck0, 1, .code);
2
2 ! Now look for either an RT11CTX block or a DOS11CTX block
2 !
2 IF .ctx [ctx$b_type] EQL exchblk$k_rt11ctx
```

```
2 THEN
3     BEGIN
3     IF .ctx [ctx$w_size] NEQ exchblk$s_rt11ctx
3     THEN
4         BEGIN
4         size = exchblk$s_rt11ctx;
4         type = exchblk$k_rt11ctx;
4         $exch_signal_stop (exch$_blockcheck, 6, .code, .ctx, .ctx [ctx$w_size], .size, .ctx [ctx$b_type], .t
4         END;
3     END
2 ELSE IF .ctx [ctx$b_type] EQL exchblk$k_dos11ctx
2 THEN
3     BEGIN
3     IF .ctx [ctx$w_size] NEQ exchblk$s_dos11ctx
3     THEN
4         BEGIN
4         size = exchblk$s_dos11ctx;
4         type = exchblk$k_dos11ctx;
4         $exch_signal_stop (exch$_blockcheck, 6, .code, .ctx, .ctx [ctx$w_size], .size, .ctx [ctx$b_type], .t
4         END;
3     END
2 ELSE
3     BEGIN
3     size = exchblk$s_rt11ctx;
3     type = exchblk$k_rt11ctx;
3     $exch_signal_stop (exch$_blockcheck, 6, .code, .ctx, .ctx [ctx$w_size], .size, .ctx [ctx$b_type], .type)
3     END;
2
2 IF .filb EQL 0
2 THEN
2     $exch_signal_stop (exch$_blockcheck0, 1, (10000+.code));
2
2 IF   .filb [filb$w_size] NEQ exchblk$s_filb
2   OR
2      .filb [filb$b_type] NEQ exchblk$k_filb
2 THEN
P 2     $exch_signal_stop (exch$_blockcheck, 6, (10000+.code), .filb,
P 2         .filb [filb$w_size], exchblk$s_filb,
2         .filb [filb$b_type], exchblk$k_filb);
2
2 IF .volb EQL 0
2 THEN
2     $exch_signal_stop (exch$_blockcheck0, 1, (20000+.code));
2
2 IF   .volb [volb$w_size] NEQ exchblk$s_volb
2   OR
2      .volb [volb$b_type] NEQ exchblk$k_volb
2 THEN
P 2     $exch_signal_stop (exch$_blockcheck, 6, (20000+.code), .volb,
P 2         .volb [volb$w_size], exchblk$s_volb,
2         .volb [volb$b_type], exchblk$k_volb);
2
1 END;


                                                        .EXTRN  EXCH$_BLOCKCHECK0
```

```
                                                      H  9
EXCH$PDP        Small PDP-11 record structure routines       16-Sep-1984 01:11:46    VAX-11 Bliss-32 V4.0-742             Page  21
V04-000         pdp_check_ctx                                14-Sep-1984 12:29:07    [EXCHNG.SRC]EXCPDP.B32;1                   (7)

                                                          .EXTRN  LIB$STOP, EXCH$_BLOCKCHECK

                                    001C 00000            .ENTRY  PDP_CHECK_CTX, Save R2,R3,R4                    ; 0729
              54 00000000G   00   9E 00002            MOVAB   LIB$STOP, R4                                            ;
              51          04 AC   D0 00009            MOVL    CTX, R1                                                 ; 0771
                             05   12 0000D            BNEQ    1$                                                      ; 0779
                          08 AC   DD 0000F            PUSHL   CODE                                                    ; 0781
                             52   11 00012            BRB     6$                                                      ;
              53          0A A1   9A 00014 1$:        MOVZBL  10(R1), R3                                              ; 0785
          F4  8F             53   91 00018            CMPB    R3, #244                                                ;
                             0A   12 0001C            BNEQ    2$                                                      ;
        0082  8F          08 A1   B1 0001E            CMPW    8(R1), #130                                             ; 0788
                             31   13 00024            BEQL    5$                                                      ;
                             18   11 00026            BRB     3$                                                      ; 0791
          FC  8F             53   91 00028 2$:        CMPB    R3, #252                                                ; 0796
                             12   12 0002C            BNEQ    3$                                                      ;
        008A  8F          08 A1   B1 0002E            CMPW    8(R1), #138                                             ; 0799
                             21   13 00034            BEQL    5$                                                      ;
              52          8A 8F   9A 00036            MOVZBL  #138, SIZE                                              ; 0802
              50          FC 8F   9A 0003A            MOVZBL  #252, TYPE                                              ; 0803
                             08   11 0003E            BRB     4$                                                      ; 0804
              52          82 8F   9A 00040 3$:        MOVZBL  #130, SIZE                                              ; 0809
              50          F4 8F   9A 00044            MOVZBL  #244, TYPE                                              ; 0810
                             50   DD 00048 4$:        PUSHL   TYPE                                                    ; 0811
                             0C   BB 0004A            PUSHR   #^M<R2,R3>                                              ;
              7E          08 A1   3C 0004C            MOVZWL  8(R1), -(SP)                                            ;
                             51   DD 00050            PUSHL   R1                                                      ;
                          08 AC   DD 00052            PUSHL   CODE                                                    ;
                             3C   11 00055            BRB     9$                                                      ;
              50          10 A1   D0 00057 5$:        MOVL    16(R1), R0                                              ; 0814
                             0B   12 0005B            BNEQ    7$                                                      ;
    7E    08  AC 00002710    8F   C1 0005D            ADDL3   #10000, CODE, -(SP)                                     ; 0816
                             3C   11 00066 6$:        BRB     11$                                                     ;
        035B  8F          08 A0   B1 00068 7$:        CMPW    8(R0), #859                                             ; 0818
                             07   12 0006E            BNEQ    8$                                                      ;
          FA  8F          0A A0   91 00070            CMPB    10(R0), #250                                            ; 0820
                             1E   13 00075            BEQL    10$                                                     ;
              7E          FA 8F   9A 00077 8$:        MOVZBL  #250, -(SP)                                             ; 0824
              7E          0A A0   9A 0007B            MOVZBL  10(R0), -(SP)                                           ;
              7E        035B 8F   3C 0007F            MOVZWL  #859, -(SP)                                             ;
              7E          08 A0   3C 00084            MOVZWL  8(R0), -(SP)                                            ;
                             50   DD 00088            PUSHL   R0                                                      ;
    7E    08  AC 00002710    8F   C1 0008A            ADDL3   #10000, CODE, -(SP)                                     ;
                             46   11 00093 9$:        BRB     14$                                                     ;
              50          14 A1   D0 00095 10$:       MOVL    20(R1), R0                                              ; 0826
                             15   12 00099            BNEQ    12$                                                     ;
    7E    08  AC 00004E20    8F   C1 0009B            ADDL3   #20000, CODE, -(SP)                                     ; 0828
                             01   DD 000A4 11$:       PUSHL   #1                                                      ;
                 00000000G   8F   DD 000A6            PUSHL   #EXCH$_BLOCKCHECK0                                      ;
              64             03   FB 000AC            CALLS   #3, LIB$STOP                                            ;
                             04      000AF            RET                                                             ;
        041B  8F          08 A0   B1 000B0 12$:       CMPW    8(R0), #1051                                            ; 0830
                             07   12 000B6            BNEQ    13$                                                     ;
          F3  8F          0A A0   91 000B8            CMPB    10(R0), #243                                            ; 0832
                             27   13 000BD            BEQL    15$                                                     ;
              7E          F3 8F   9A 000BF 13$:       MOVZBL  #243, -(SP)                                             ; 0836
              7E          0A A0   9A 000C3            MOVZBL  10(R0), -(SP)                                           ;
```

```
                              7E      041B   8F   3C 000C7          MOVZWL  #1051, -(SP)                         ;
                              7E        08   A0   3C 000CC          MOVZWL  8(R0), -(SP)                         ;
                                             50   DD 000D0          PUSHL   R0                                   ;
                    7E      08   AC 00004E20 8F   C1 000D2          ADDL3   #20000, CODE, -(SP)                  ;
                                             06   DD 000DB 14$:     PUSHL   #6                                   ;
                                 00000000G   8F   DD 000DD          PUSHL   #EXCH$_BLOCKCHECK                    ;
                              64             08   FB 000E3          CALLS   #8, LIB$STOP                         ;
                                                  04 000E6 15$:     RET                                          ; 0838

; Routine Size:  231 bytes,    Routine Base:  EXCH$PDP_CODE + 0378
```

```
1 GLOBAL ROUTINE pdp_copy_binary_record (in_len, in_buf : $ref_bvector,    %SBTTL 'pdp_copy_binary_record'
1                                         out_buf : $ref_bvector) : NOVALUE =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Copy the input record to a buffer, reformatting it as a valid formatted-binary record.
2 !
2 ! INPUTS:
2 !
2 !       in_len  - length of the input record
2 !       in_buf  - address of the input record
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       out_buf - address of the output buffer which receives the formatted-binary copy of the input
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('pdp_copy_binary_record> ');
2
2 REGISTER
2     ip,                                        ! Input pointer
2     op,                                        ! Output pointer
2     chksum      : BYTE,
2     neg_chksum  : BYTE,
2     char        : BYTE                         ! Current character
2     ;
2
2 BIND
2     sentinel = out_buf [0] : WORD,             ! Sentinel word, first two bytes of the output
2     length   = out_buf [2] : WORD              ! Length word, next two bytes
2     ;
2
2 $debug_print_fao ('entry, len=!UL, buf[0:19]="!AF"', .in_len, 20, .in_buf);
2
2 ! Initialize our local data segments
2 !
2 op    = .out_buf;                                       ! Output buffer pointer
2 ip    = .in_buf;                                        ! Input pointer at the start of the record
2 chksum = 0;
```

```
;    807        0896  2
;    808        0897  2  ! Put the sentinel and length words in the buffer
;    809        0898  2  !
;    810        0899  2  sentinel = 1;
;    811        0900  2  length = .in_len + 4;
;    812        0901  2
;    813        0902  2  ! Prepare the checksum from the first four bytes
;    814        0903  2  !
;    815        0904  2  DECR c FROM 3 TO 0
;    816        0905  2  DO
;    817        0906  2      chksum = .chksum + CH$RCHAR_A (op);
;    818        0907  2
;    819        0908  2  ! Start grabbing bytes
;    820        0909  2  !
;    821        0910  2  IF .in_len GTRU 0
;    822        0911  2  THEN
;    823        0912  2      DECR c FROM .in_len-1 TO 0
;    824        0913  2      DO
;    825        0914  3          BEGIN
;    826        0915  3
;    827        0916  3          char = CH$RCHAR_A (ip);                ! Read the new character and advance the input pointer
;    828        0917  3
;    829        0918  3          chksum = .chksum + .char;              ! Add this byte to the checksum
;    830        0919  3
;    831        0920  3          CH$WCHAR_A (.char, op);                ! Move it to the output and advance the output pointer
;    832        0921  3
;    833        0922  2          END;
;    834        0923  2
;    835        0924  2  ! Store the negated checksum
;    836        0925  2  !
;    837        0926  2  neg_chksum = -.chksum;
;    838        0927  2  CH$WCHAR (.neg_chksum, .op);               ! Move it to the output
;    839        0928  2
;    840        0929  2  RETURN;
;    841        0930  1  END;



                                          001C 00000        .ENTRY  PDP_COPY_BINARY_RECORD, Save R2,R3,R4        ; 0839
          53      0C   AC            02   C1 00002        ADDL3   #2, OUT_BUF, R3                                 ; 0886
                       50      08    AC   7D 00007        MOVQ    IN_BUF, IP                                      ; 0894
                                     52   94 0000B        CLRB    CHKSUM                                          ; 0895
                  0C   BC            01   B0 0000D        MOVW    #1, @OUT_BUF                                    ; 0899
          63      04   AC            04   A1 00011        ADDW3   #4, IN_LEN, (R3)                                ; 0900
                       53            03   D0 00016        MOVL    #3, C                                           ; 0904
                       54            81   9A 00019 1$:    MOVZBL  (OP)+, R4                                       ; 0906
                       52            54   80 0001C        ADDB2   R4, CHKSUM                                      ;
                       F7            53   F4 0001F        SOBGEQ  C, 1$                                           ;
                               04    AC   D5 00022        TSTL    IN_LEN                                          ; 0910
                                     12   13 00025        BEQL    4$                                              ;
                  54           04    AC   D0 00027        MOVL    IN_LEN, C                                       ; 0912
                                     09   11 0002B        BRB     3$                                              ;
                       53            80   90 0002D 2$:    MOVB    (IP)+, CHAR                                     ; 0916
                       52            53   80 00030        ADDB2   CHAR, CHKSUM                                    ; 0918
                       81            53   90 00033        MOVB    CHAR, (OP)+                                     ; 0920
```

```
                                F4          54  F4 00036 3$:    SOBGEQ  C, 2$                                                   ; 0912
                                50          52  8E 00039 4$:    MNEGB   CHKSUM, NEG_CHKSUM                                      ; 0926
                                61          50  90 0003C        MOVB    NEG_CHKSUM, (OP)                                        ; 0927
                                                04 0003F        RET                                                             ; 0930

; Routine Size:  64 bytes,    Routine Base:  EXCH$PDP_CODE + 045F
```

```
1 GLOBAL ROUTINE pdp_copy_stream_record (in_len, in_buf : $ref_bvector, %SBTTL 'pdp_copy_stream_record'
1                                          out_buf : $ref_bvector) =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Copy the input record to a buffer, reformatting it as a valid stream format record.  The length of t
2 !       output record is returned.
2 !
2 ! INPUTS:
2 !
2 !       in_len  - length of the input record
2 !       in_buf  - address of the input record
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       out_buf - address of the output buffer which receives the stream format copy of the input, including
2 !                 record terminator(s)
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       The length of the output record, including terminator
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('pdp_copy_stream_record> ');
2
2 REGISTER
2     ip,                                       ! Input pointer
2     op,                                       ! Output pointer
2     ol,                                       ! Output length
2     char          : BYTE                      ! Current character
2     ;
2
2 $debug_print_fao ('entry, len=!UL, buf[0:19]="!AF"', .in_len, 20, .in_buf);
2
2 ! Initialize our local data segments
2 !
2 op   = .out_buf;                                      ! Output buffer pointer
2 ip   = .in_buf;                                       ! Input pointer at the start of the record
2 char = 0;                                             ! Preset for the later test, in case 0 length input
2
2 ! Start grabbing bytes
2 !
2 IF .in_len GTRU 0
```

```
                                                           N 9
EXCH$PDP            Small PDP-11 record structure routines         16-Sep-1984 01:11:46    VAX-11 Bliss-32 V4.0-742            Page  27
V04-000             pdp_copy_stream_record                          14-Sep-1984 12:29:07    [EXCHNG.SRC]EXCPDP.B32;1                 (9)

;   900          0988  2 THEN
;   901          0989  2      DECR c FROM .in_len-1 TO 0
;   902          0990  2      DO
;   903          0991  3           BEGIN
;   904          0992  3
;   905          0993  3           ! Read the character and clear the high bit
;   906          0994  3           !
;   907          0995  3           char = CH$RCHAR_A (ip);          ! Read the new character and advance the input pointer
;   908          0996  3           char <7,1,0> = 0;                ! Clear the high bit
;   909          0997  3
;   910          0998  3           ! Now look at the character and do something with it
;   911          0999  3           !
;   912          1000  3           SELECTONEU .char OF
;   913          1001  3           SET
;   914          1002  3
;   915          1003  3                [NUL, DEL, VT] :
;   916          1004  3                        ;
;   917          1005  3
;   918          1006  3                [OTHERWISE] :
;   919          1007  3                        CH$WCHAR_A (.char, op);
;   920          1008  3
;   921          1009  3           TES;
;   922          1010  3
;   923          1011  2           END;
;   924          1012  2
;   925          1013  2 ! If the final char was either a form feed or a line feed, we are done.  Otherwise add the <CR><LF> pair
;   926          1014  2 !
;   927          1015  4 IF ((.char NEQ LF)        ! line feed
;   928          1016  3   AND
;   929          1017  3      (.char NEQ FF))      ! form feed
;   930          1018  2 THEN
;   931          1019  3      BEGIN
;   932          1020  3      CH$WCHAR_A (CR, op);
;   933          1021  3      CH$WCHAR_A (LF, op);
;   934          1022  2      END;
;   935          1023  2
;   936          1024  2 ! Calculate the final length
;   937          1025  2 !
;   938          1026  2 ol = .op - .out_buf;
;   939          1027  2
;   940          1028  2 $debug_print_fao ('output len !UL, record[0:19] "!AF"', .ol, 20, .out_buf);
;   941          1029  2
;   942          1030  2 RETURN .ol;
;   943          1031  1 END;



                                       000C 00000          .ENTRY  PDP_COPY_STREAM_RECORD, Save R2,R3           ; 0931
                    50        08   AC   7D 00002          MOVQ    IN_BUF, IP                                   ; 0982
                                   52   94 00006          CLRB    CHAR                                         ; 0983
                              04   AC   D5 00008          TSTL    IN_LEN                                       ; 0987
                                   20   13 0000B          BEQL    5$                                           ;
                    53        04   AC   D0 0000D          MOVL    IN_LEN, C                                    ; 0989
                                   17   11 00011          BRB     2$                                           ;
                    52             80   90 00013 1$:      MOVB    (IP)+, CHAR                                  ; 0995
```

```
                            52      80  8F  8A 00016          BICB2   #128, CHAR                                ; 0996
                                        0E  13 0001A          BEQL    2$                                        ; 1003
                            0B          52  91 0001C          CMPB    CHAR, #11                                 ;
                                        09  13 0001F          BEQL    2$                                        ;
                        7F  8F          52  91 00021          CMPB    CHAR, #127                                ;
                                        03  13 00025          BEQL    2$                                        ;
                            81          52  90 00027          MOVB    CHAR, (OP)+                               ; 1007
                            E6          53  F4 0002A 2$:      SOBGEQ  C, 1$                                     ; 0989
                            0A          52  91 0002D 3$:      CMPB    CHAR, #10                                 ; 1015
                                        0A  13 00030          BEQL    4$                                        ;
                            0C          52  91 00032          CMPB    CHAR, #12                                 ; 1017
                                        05  13 00035          BEQL    4$                                        ;
                            81    0A0D  8F  B0 00037          MOVW    #2573, (OP)+                              ; 1020
                  50        51      0C  AC  C3 0003C 4$:      SUBL3   OUT_BUF, OP, OL                           ; 1026
                                            04 00041          RET                                               ; 1031

; Routine Size:  66 bytes,    Routine Base:  EXCH$PDP_CODE + 049F
```

```
1 GLOBAL ROUTINE exch$pdp_filter_filename (nam_len, nam_start) =  %SBTTL 'exch$pdp_filter_filename'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Scan filename, removing characters which are invalid.  The string will be modified in place.
2 !
2 ! INPUTS:
2 !
2 !       nam_len   - length of the name
2 !       nam_start - starting address of the filename
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       the name string is modified in place
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('exch$pdp_filter_filename> ');
2
2
2 REGISTER
2     ip,                               ! Input pointer
2     op,                               ! Output pointer
2     char        : BYTE                ! Current character
2     ;
2
2 $debug_print_lit ('entry');
2
3 IF (.nam_len EQL 0)                                 ! Nothing to do in this case
2 THEN
2     RETURN .nam_len;
2
2 ! Initialize our local data segments
2 !
2 ip   = .nam_start;                                  ! Input pointer at the start of the buffer
2 op   = .ip;                                         ! Output pointer starts at the beginning
2
2 DECR len FROM .nam_len - 1 TO 0
2 DO
3     BEGIN
```

```
; 1002          1089  3        char = CH$RCHAR_A (ip);
; 1003          1090  3        SELECTONE .char OF
; 1004          1091  3        SET
; 1005          1092  3            ['A' TO 'Z', '0' TO '9'] :
; 1006          1093  3                                                  CH$WCHAR_A (.char, op);
; 1007          1094  3            [OTHERWISE] :
; 1008          1095  3        TES;                                      ;
; 1009          1096  2        END;
; 1010          1097  2
; 1011          1098  2    ! Return the length
; 1012          1099  2    !
; 1013          1100  2    RETURN .op - .nam_start;
; 1014          1101  2
; 1015          1102  1    END;



                                          000C 00000           .ENTRY   EXCH$PDP_FILTER_FILENAME, Save R2,R3               ; 1032
                         53      04    AC   D0 00002           MOVL     NAM_LEN, R3                                        ; 1077
                                       04   12 00006           BNEQ     1$                                                 ;
                         50            53   D0 00008           MOVL     R3, R0                                             ; 1079
                                            04 0000B           RET                                                         ;
                         50      08    AC   D0 0000C 1$:       MOVL     NAM_START, IP                                      ; 1083
                         51            50   D0 00010           MOVL     IP, OP                                             ; 1084
                                       1C   11 00013           BRB      5$                                                 ; 1086
                         52            80   90 00015 2$:       MOVB     (IP)+, CHAR                                        ; 1089
                         30            52   91 00018           CMPB     CHAR, #48                                          ; 1092
                                       05   1F 0001B           BLSSU    3$                                                 ;
                         39            52   91 0001D           CMPB     CHAR, #57                                          ;
                                       0C   1B 00020           BLEQU    4$                                                 ;
                    41   8F            52   91 00022 3$:       CMPB     CHAR, #65                                          ;
                                       09   1F 00026           BLSSU    5$                                                 ;
                    5A   8F            52   91 00028           CMPB     CHAR, #90                                          ;
                                       03   1A 0002C           BGTRU    5$                                                 ;
                         81            52   90 0002E 4$:       MOVB     CHAR, (OP)+                                        ; 1093
                         E1            53   F4 00031 5$:       SOBGEQ   LEN, 2$                                            ; 1086
                         51      08    AC   C2 00034           SUBL2    NAM_START, R1                                      ; 1100
                         50            51   D0 00038           MOVL     R1, R0                                             ;
                                            04 0003B           RET                                                         ; 1102

; Routine Size:  60 bytes,    Routine Base:  EXCH$PDP_CODE + 04E1
```

```
1 GLOBAL ROUTINE pdp_find_binary_record (filb : $ref_bblock, buf_start, %SBTTL 'pdp_find_binary_record'
1                     buf_end : $ref_bvector, new_start) =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Scan buffer from start to end (if necessary) looking for a single formatted binary record.  The addr
2 !       length of the record are copied to the record buffer pointers in the filb.  The address of the next
2 !       unscanned byte is returned.
2 !
2 ! INPUTS:
2 !
2 !       filb      - pointer to the filb which contains the active record stream
2 !       buf_start - starting address in buffer to scan
2 !       buf_end   - one past the highest valid buffer address
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       new_start - receives address of first unscanned byte
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       findbin$k_success       - record 'placed' in filb, all is well
2 !               k_eob           - at end of buffer without finding complete record
2 !               k_bad_fmt       - problem with record format
2 !               k_too_big       - record exceeds length of output buffer
2 !               k_chksum        - computed checksum differs from stored checksum
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('pdp_find_binary_record> ');
2
2
2 REGISTER
2     ip,                                 ! Input pointer
2     ol,                                 ! Output length
2     eob,                                ! End of buffer
2     chksum      : BYTE,                 ! Check sum accumulator
2     neg_chksum  : BYTE,                 ! Negative of checksum for compares
2     char        : BYTE                  ! Current character
2     ;
2
2 $debug_print_lit ('entry');
2 $block_check (2, .filb, filb, 495);
2
```

```
2 ! Initialize our local data segments
2 !
2 ip   = .buf_start;                                  ! Input pointer at the start of the buffer
2 eob  = .buf_end;                                    ! End of buffer pointer one past the end of the buffer
2
2 ! Skip any null bytes at the start of the record
2 !
2 DO
3     BEGIN
3
3     ! Check for the end of the input buffer.  We make sure that the entire header is in the buffer
3     !
3     IF .ip+4 GEQU .eob
3     THEN
3         RETURN findbin$k_eob;
3
3     ! Read the character and advance the pointer
3     !
3     char = CH$RCHAR_A (ip);
3
3     END
3
2 UNTIL .char NEQ 0;
2
2 ! A formatted binary record has a word containing 1 followed by a word containing the length of the data + h
2 !
3 IF (.char NEQ 1) OR (CH$RCHAR_A (ip) NEQ 0)
2 THEN
2     RETURN findbin$k_bad_fmt;
2
2 ! Get the length, and initialize the checksum
2 !
2 ol = (BIND len = .ip : WORD; .len) - 4;          ! Interpret datum at input pointer as a word
2 chksum = 1 + CH$RCHAR_A (ip) + CH$RCHAR_A (ip); ! Checksum is 1 plus the two bytes of the length word
2
2 ! Although we use locate mode, lets do a sanity check and refuse oversize records
2 !
2 IF .ol GTRU filb$s_record_buffer
2 THEN
2     RETURN findbin$k_too_big;
2
2 ! Make sure that the entire record plus the checksum byte are present in the buffer
2 !
2 IF (.ip + .ol + 1) GEQU .eob
2 THEN
2     RETURN findbin$k_eob;
2
2 ! Point the filb record information at the record we have found
2 !
2 filb [filb$a_record] = .ip;
2 filb [filb$l_record_len] = .ol;
2
2 ! Calculate the checksum, then negate it
2 !
2 DECR count FROM .ol-1 TO 0 DO chksum = .chksum + CH$RCHAR_A (ip);
2 neg_chksum = -.chksum;
2
```

```
; 1131           1217  2  ! Get the stored checksum from the end of the record
; 1132           1218  2  !
; 1133           1219  2  char = CH$RCHAR_A (ip);                              ! Get the stored checksum
; 1134           1220  2  .new_start = .ip;                                    ! Send back the start of the next record
; 1135           1221  2
; 1136           1222  2  IF .neg_chksum NEQ .char
; 1137           1223  2  THEN
; 1138           1224  3      BEGIN
; 1139           1225  3
; 1140           1226  3      ! The RSX/VMS utility FLX has been calculating incorrect checksums for records longer than 255 bytes.  I
; 1141           1227  3      ! to include the high order byte of the length in the checksum.  If the checksum is correct when we assu
; 1142           1228  3      ! that this has occurred, accept it as correct.
; 1143           1229  3      !
; 1144           1230  3      $debug_print_fao ('Record length !UL, checksum !OB, calc chksum !OB', .ol, .char, .neg_chksum);
; 1145           1231  3      chksum = .chksum - ((.ol+4) / 256);           ! Pretend we never added the high byte
; 1146           1232  3      neg_chksum = -.chksum;
; 1147           1233  3      IF .neg_chksum NEQ .char
; 1148           1234  3      THEN
; 1149           1235  4          BEGIN
; 1150           1236  4          $debug_print_fao ('Record length !UL, checksum !OB, calc chksum !OB', .ol, .char, .neg_chksum);
; 1151           1237  4          RETURN findbin$k_chksum;
; 1152           1238  3          END;
; 1153           1239  2      END;
; 1154           1240  2
; 1155           1241  2  RETURN findbin$k_success;
; 1156           1242  1  END;



                                                                .EXTRN  EXCH$UTIL_BLOCK_CHECK

                                             01FC 00000         .ENTRY  PDP_FIND_BINARY_RECORD, Save R2,R3,R4,R5,-  ; 1103
                                                                        R6,R7,R8
                         57        04    AC   D0 00002          MOVL    FILB, R7                                    ; 1158
                         52 035B00FA     8F   D0 00006          MOVL    #56295674, R2
                         51      01EF    8F   3C 0000D          MOVZWL  #495, R1
                         50                 57   D0 00012       MOVL    R7, R0
                            00000000G    EF   16 00015          JSB     EXCH$UTIL_BLOCK_CHECK
                         50        08    AC   D0 0001B          MOVL    BUF_START, IP                               ; 1162
                         52        0C    AC   D0 0001F          MOVL    BUF_END, EOB                                ; 1163
                         51        04    A0   9E 00023 1$:      MOVAB   4(R0), R1                                   ; 1172
                         52                 51   D1 00027       CMPL    R1, EOB                                     ;
                                            3E   1E 0002A       BGEQU   5$                                          ;
                         54                 80   90 0002C       MOVB    (IP)+, CHAR                                 ; 1178
                                            F2   13 0002F       BEQL    1$                                          ; 1182
                         01                 54   91 00031       CMPB    CHAR, #1                                    ; 1186
                                            05   12 00034       BNEQ    2$
                         51                 80   9A 00036       MOVZBL  (IP)+, R1
                                            04   13 00039       BEQL    3$
                         50                 04   D0 0003B 2$:   MOVL    #4, R0                                      ; 1188
                                                 04 0003E       RET
                         51                 60   3C 0003F 3$:   MOVZWL  (IP), OL                                    ; 1192
                         51                 04   C2 00042       SUBL2   #4, OL
                         55                 80   9A 00045       MOVZBL  (IP)+, R5                                   ; 1193
                         56                 80   9A 00048       MOVZBL  (IP)+, R6
                         58         01 A645   9E 0004B          MOVAB   1(R6)[R5], R8
                         53                 58   90 00050       MOVB    R8, CHKSUM                                  ;
```

```
                    00000200   8F          51  D1 00053       CMPL    OL, #512                        ; 1197
                                           04  1B 0005A       BLEQU   4$
                               50          03  D0 0005C       MOVL    #3, R0                          ; 1199
                                               04 0005F       RET
                               56     01 A140  9E 00060 4$:   MOVAB   1(OL)[IP], R6                   ; 1203
                               52          56  D1 00065       CMPL    R6, EOB
                                           04  1F 00068       BLSSU   6$
                               50          01  D0 0006A 5$:   MOVL    #1, R0                          ; 1205
                                               04 0006D       RET
                         46    A7          50  D0 0006E 6$:   MOVL    IP, 70(R7)                      ; 1209
                         42    A7          51  D0 00072       MOVL    OL, 66(R7)                      ; 1210
                               52          51  D0 00076       MOVL    OL, COUNT                       ; 1214
                                           06  11 00079       BRB     8$
                               55          80  9A 0007B 7$:   MOVZBL  (IP)+, R5
                               53          55  80 0007E       ADDB2   R5, CHKSUM
                               F7          52  F4 00081 8$:   SOBGEQ  COUNT, 7$
                               52          53  8E 00084       MNEGB   CHKSUM, NEG_CHKSUM              ; 1215
                               54          80  90 00087       MOVB    (IP)+, CHAR                     ; 1219
                         10    BC          50  D0 0008A       MOVL    IP, @NEW_START                  ; 1220
                               54          52  91 0008E       CMPB    NEG_CHKSUM, CHAR                ; 1222
                                           19  13 00091       BEQL    9$
                               51          04  C0 00093       ADDL2   #4, R1                          ; 1231
                               51  00000100 8F C6 00096       DIVL2   #256, R1
                               53          51  82 0009D       SUBB2   R1, CHKSUM
                               52          53  8E 000A0       MNEGB   CHKSUM, NEG_CHKSUM              ; 1232
                               54          52  91 000A3       CMPB    NEG_CHKSUM, CHAR                ; 1233
                                           04  13 000A6       BEQL    9$
                               50          02  D0 000A8       MOVL    #2, R0                          ; 1237
                                               04 000AB       RET
                                           50  D4 000AC 9$:   CLRL    R0                              ; 1241
                                               04 000AE       RET                                     ; 1242

; Routine Size:  175 bytes,    Routine Base:  EXCH$PDP_CODE + 051D
```

```
GLOBAL ROUTINE pdp_find_stream_record (filb : $ref_bblock, buf_start, %SBTTL 'pdp_find_stream_record'
                buf_end : $ref_bvector, new_start) =
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Scan buffer from start to end (if necessary) looking for a single stream record.  The reformatted
!       record is copied to the record buffer in the filb.  The address of the next unscanned byte is return
!
! INPUTS:
!
!       buf_start - starting address in buffer to scan
!       buf_end   - one past the highest valid buffer address
!       filb      - pointer to the filb which contains the active record stream
!
! IMPLICIT INPUTS:
!
!       none
!
! OUTPUTS:
!
!       new_start - receives address of first unscanned byte
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       findstm$k_success       - record placed in filb, all is well
!         ...   k_ctrlz_eof     - ^Z at start of record
!               k_eob           - at end of buffer, no record found
!               k_no_term       - reached end of buffer in middle of record
!               k_bad_fmt       - record exceeds length of output buffer
!
! SIDE EFFECTS:
!
!       none
!--

$dbgtrc_prefix ('pdp_find_stream_record> ');

LOCAL
    status
    ;

REGISTER
    ip,                                 ! Input pointer
    op,                                 ! Output pointer
    ol,                                 ! Output length
    eob,                                ! End of buffer
    char        : BYTE                  ! Current character
    ;

$debug_print_lit ('entry');
$block_check (2, .filb, filb, 429);
```

```
2
2   ! Set address of the filb record to the start of the filb record buffer
2   !
2   filb [filb$a_record]     = filb [filb$t_record_buffer];
2
2   ! Initialize our local data segments
2   !
2   op   = filb [filb$t_record_buffer];          ! Output pointer to the filb buffer
2   ol   = 0;                                     ! Output length starts at zero
2   ip   = .buf_start;                            ! Input pointer at the start of the buffer
2   eob  = .buf_end;                              ! End of buffer pointer one past the end of the buffer
2   status = findstm$k_success;
2
2   ! Start grabbing bytes
2   !
2   $debug_print_fao ('ip !XL, eob !XL, ol !XW, char "!AF"', .ip, .eob, .ol, 1, .ip);
2   WHILE 1
2   DO
3       BEGIN
3
3       ! Check for the end of either of the buffers
3       !
3       IF .ip GEQU .eob                          ! If the input pointer is past the end of the input buffer
3       THEN
4           BEGIN
4           IF .ol EQL 0                          ! If the output length is still zero
4           THEN
4               status = findstm$k_eob            !  then end-of-buffer without any record
4           ELSE
4               status = findstm$k_no_term;       !  otherwise record without terminator
4           EXITLOOP;
4           END;
3
3       IF .ol GTRU filb$s_record_buffer          ! If the output length is gtr than the buffer (the buffer ac
3       THEN                                      !  has an extra guard byte at the end so no overrun problem)
4           BEGIN
4           status = findstm$k_bad_fmt;           ! Our status is bad format record
4           EXITLOOP;
4           END;
3
3       ! Read the character and clear the high bit
3       !
3       char = CH$RCHAR_A (ip);            ! Read the new character and advance the input pointer
3       char <7,1,0> = 0;                  ! Clear the high bit
3
3       ! Now look at the character and do something with it
3       !
3       SELECTONEU .char OF
3       SET
3
3           [NUL, DEL, VT] :
3               ;
3
3           [CTRLZ] :                             ! Control/z marks end of file if the first char
4               BEGIN
4               IF .ol EQL 0
4               THEN
```

```
; 1272       1357  5                     BEGIN
; 1273       1358  5                     status = findstm$k_ctrlz_eof;        ! Fine, no record
; 1274       1359  5                     EXITLOOP;
; 1275       1360  5                     END
; 1276       1361  4                 ELSE
; 1277       1362  5                     BEGIN
; 1278       1363  5                     CH$WCHAR_A (.char, op);
; 1279       1364  5                     ol = .ol + 1;
; 1280       1365  4                     END;
; 1281       1366  3                 END;
; 1282       1367  3
; 1283       1368  3         [FF] :
; 1284       1369  4                 BEGIN
; 1285       1370  4                 CH$WCHAR_A (.char, op);
; 1286       1371  4                 ol = .ol + 1;
; 1287       1372  4                 EXITLOOP;
; 1288       1373  3                 END;
; 1289       1374  3
; 1290       1375  3         [LF] :
; 1291       1376  4                 BEGIN
; 1292       1377  4                 IF .ol GTRU 0
; 1293       1378  4                 THEN
; 1294       1379  5                     BEGIN
; 1295       1380  5                     IF CH$RCHAR (.op-1) EQL cr
; 1296       1381  5                     THEN
; 1297       1382  5                         ol = .ol - 1;
; 1298       1383  4                     END;
; 1299       1384  4                 EXITLOOP;
; 1300       1385  3                 END;
; 1301       1386  3
; 1302       1387  3         [OTHERWISE] :
; 1303       1388  4                 BEGIN
; 1304       1389  4                 CH$WCHAR_A (.char, op);
; 1305       1390  4                 ol = .ol + 1;
; 1306       1391  3                 END;
; 1307       1392  3
; 1308       1393  3     TES;
; 1309       1394  3
; 1310       1395  2     END;
; 1311       1396  2
; 1312       1397  2 .new_start = .ip;
; 1313       1398  2 filb [filb$l_record_len] = .ol;
; 1314       1399  2
; 1315       1400  2 $debug_print_fao ('record "!AF", len !UL, status !UL', .ol, filb [filb$t_record_buffer], .ol, .status);
; 1316       1401  2
; 1317       1402  2 RETURN .status;
; 1318       1403  1 END;



                                    007C 00000          .ENTRY  PDP_FIND_STREAM_RECORD, Save R2,R3,R4,R5,R6 ; 1243
                   56        04  AC   D0 00002          MOVL    FILB, R6                                      ; 1299
                   52 035B00FA  8F   D0 00006          MOVL    #56295674, R2                                 ;
                   51      01AD  8F   3C 0000D          MOVZWL  #429, R1                                      ;
                   50            56   D0 00012          MOVL    R6, R0                                        ;
```

```
                                                                      L 10
EXCH$PDP        Small PDP-11 record structure routines           16-Sep-1984 01:11:46     VAX-11 Bliss-32 V4.0-742          Page  38
V04-000         pdp_find_stream_record                           14-Sep-1984 12:29:07     [EXCHNG.SRC]EXCPDP.B32;1               (12)

                            00000000G  EF  16 00015          JSB     EXCH$UTIL_BLOCK_CHECK                      ;
                   50           015A   C6  9E 0001B          MOVAB   346(R6), R0                                ; 1303
             46    A6                  50  D0 00020          MOVL    R0, 70(R6)                                 ;
                   51                  50  D0 00024          MOVL    R0, OP                                     ; 1307
                                       52  D4 00027          CLRL    OL                                         ; 1308
                   50             08   AC  D0 00029          MOVL    BUF_START, IP                              ; 1309
                   53             0C   AC  D0 0002D          MOVL    BUF_END, EOB                               ; 1310
                                       55  D4 00031          CLRL    STATUS                                     ; 1311
                   53                  50  D1 00033 1$:      CMPL    IP, EOB                                    ; 1322
                                       0E  1F 00036          BLSSU   3$                                         ;
                                       52  D5 00038          TSTL    OL                                         ; 1325
                                       05  12 0003A          BNEQ    2$                                         ;
                   55                  02  D0 0003C          MOVL    #2, STATUS                                 ; 1327
                                       5B  11 0003F          BRB     8$                                         ;
                   55                  03  D0 00041 2$:      MOVL    #3, STATUS                                 ; 1329
                                       56  11 00044          BRB     8$                                         ; 1324
       00000200    8F                  52  D1 00046 3$:      CMPL    OL, #512                                   ; 1333
                                       05  1B 0004D          BLEQU   4$                                         ;
                   55                  04  D0 0004F          MOVL    #4, STATUS                                 ; 1336
                                       48  11 00052          BRB     8$                                         ; 1335
                   54                  80  90 00054 4$:      MOVB    (IP)+, CHAR                                ; 1342
                   54             80   8F  8A 00057          BICB2   #128, CHAR                                 ; 1343
                                       D6  13 0005B          BEQL    1$                                         ; 1350
                   0B                  54  91 0005D          CMPB    CHAR, #11                                  ;
                                       D1  13 00060          BEQL    1$                                         ;
             7F    8F                  54  91 00062          CMPB    CHAR, #127                                 ;
                                       CB  13 00066          BEQL    1$                                         ;
                   1A                  54  91 00068          CMPB    CHAR, #26                                  ; 1353
                                       09  12 0006B          BNEQ    5$                                         ;
                                       52  D5 0006D          TSTL    OL                                         ; 1355
                                       24  12 0006F          BNEQ    7$                                         ;
                   55                  01  D0 00071          MOVL    #1, STATUS                                 ; 1358
                                       26  11 00074          BRB     8$                                         ; 1357
                   0C                  54  91 00076 5$:      CMPB    CHAR, #12                                  ; 1368
                                       07  12 00079          BNEQ    6$                                         ;
                   81                  54  90 0007B          MOVB    CHAR, (OP)+                                ; 1370
                                       52  D6 0007E          INCL    OL                                         ; 1371
                                       1A  11 00080          BRB     8$                                         ; 1370
                   0A                  54  91 00082 6$:      CMPB    CHAR, #10                                  ; 1375
                                       0E  12 00085          BNEQ    7$                                         ;
                                       52  D5 00087          TSTL    OL                                         ; 1377
                                       11  13 00089          BEQL    8$                                         ;
                   0D             FF   A1  91 0008B          CMPB    -1(OP), #13                                ; 1380
                                       08  12 0008F          BNEQ    8$                                         ;
                                       52  D7 00091          DECL    OL                                         ; 1382
                                       07  11 00093          BRB     8$                                         ; 1376
                   81                  54  90 00095 7$:      MOVB    CHAR, (OP)+                                ; 1389
                                       52  D6 00098          INCL    OL                                         ; 1390
                                       97  11 0009A          BRB     1$                                         ; 1316
             10    BC                  50  D0 0009C 8$:      MOVL    IP, @NEW_START                             ; 1397
             42    A6                  52  D0 000A0          MOVL    OL, 66(R6)                                 ; 1398
                   50                  55  D0 000A4          MOVL    STATUS, R0                                 ; 1402
                                           04 000A7          RET                                                ; 1403

; Routine Size:  168 bytes,    Routine Base:  EXCH$PDP_CODE + 05CC
```

```
: 1320      1404  1 GLOBAL ROUTINE exch$pdp_flush_write_buffer (ctx : $ref_bblock) =           %SBTTL 'exch$pdp_flush_write_buffer
: 1321      1405  2 BEGIN
: 1322      1406  2 !++
: 1323      1407  2 ! FUNCTIONAL DESCRIPTION:
: 1324      1408  2 !
: 1325      1409  2 !       External entry to call buffer flush routine
: 1326      1410  2 !
: 1327      1411  2 ! INPUTS:
: 1328      1412  2 !
: 1329      1413  2 !       ctx - ctx pointer to context for an open RT11 file
: 1330      1414  2 !
: 1331      1415  2 ! IMPLICIT INPUTS:
: 1332      1416  2 !
: 1333      1417  2 !       none
: 1334      1418  2 !
: 1335      1419  2 ! OUTPUTS:
: 1336      1420  2 !
: 1337      1421  2 !       none
: 1338      1422  2 !
: 1339      1423  2 ! IMPLICIT OUTPUTS:
: 1340      1424  2 !
: 1341      1425  2 !       none
: 1342      1426  2 !
: 1343      1427  2 ! ROUTINE VALUE:
: 1344      1428  2 !
: 1345      1429  2 !       true if success, false if any error
: 1346      1430  2 !
: 1347      1431  2 ! SIDE EFFECTS:
: 1348      1432  2 !
: 1349      1433  2 !       error conditions will be signaled
: 1350      1434  2 !--
: 1351      1435  2 $dbgtrc_prefix ('pdp_flush_write_buffer> ');
: 1352      1436  2
: 1353      1437  2 LOCAL
: 1354      1438  2     status
: 1355      1439  2     ;
: 1356      1440  2
: 1357      1441  2 $debug_print_lit ('entry');
: 1358      1442  2
: 1359      1443  2 $check_call (3, pdp_check_ctx, .ctx, 455);                ! $block_check (2, .ctx, (dos11ctx or rt11ctx), 455)
: 1360      1444  2
: 1361      1445  2 ctx [ctx$v_flush] = true;                     ! Tells advance routine to flush the last block
: 1362      1446  2 status = pdp_buffer_advance_write (.ctx);      ! Flush any blocks that are sitting in the output buffer
: 1363      1447  2 ctx [ctx$v_flush] = false;                    ! Clear the flush flag
: 1364      1448  2
: 1365      1449  2 RETURN .status;
: 1366      1450  1 END;



                               0004 00000         .ENTRY  EXCH$PDP_FLUSH_WRITE_BUFFER, Save R2            ; 1404
            52       04     AC   D0 00002         MOVL    CTX, R2                                         ; 1445
       28   A2              04   88 00006         BISB2   #4, 40(R2)                                      ;
                            52   DD 0000A         PUSHL   R2                                              ; 1446
     FAEA   CF              01   FB 0000C         CALLS   #1, PDP_BUFFER_ADVANCE_WRITE                    ;
```

```
                              28   A2            04  8A 00011          BICB2   #4, 40(R2)                                   ; 1447
                                                     04 00015          RET                                                  ; 1450

; Routine Size:  22 bytes,    Routine Base:  EXCH$PDP_CODE + 0674
```

```
1 GLOBAL ROUTINE exch$pdp_get (filb : $ref_bblock) =      %SBTTL 'exch$pdp_get (filb)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Common dispatch for RT11 get routines.
2 !
2 ! INPUTS:
2 !
2 !       filb - pointer to filb for an open RT11 file
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_get> ');
2
2 LOCAL
2     buf_start,                          ! Pointer to next byte in the buffer
2     buf_end,                            ! -> one past the end of buffer
2     routn                               ! Address of action routine
2     ;
2
2 BIND
2     ctx  = filb [filb$a_context]        : $ref_bblock,
2     volb = filb [filb$a_assoc_volb]     : $ref_bblock
2     ;
```

```
; 1414          1496  2 $debug_print_lit ('entry');
; 1415          1497  2
; 1416          1498  2 $block_check (2, .filb, filb, 456);
; 1417          1499  2 $block_check (2, .volb, volb, 493);
; 1418          1500  2 $check_call (1, pdp_check_ctx, .ctx, 494);          ! $block_check (1, .ctx, (dos11ctx or rt11ctx), 494)
; 1419          1501  2 $logic_check (2, (.ctx [ctx$a_assoc_filb] EQL .filb), 134);
; 1420          1502  2 $logic_check (2, (.ctx [ctx$a_assoc_volb] EQL .volb), 135);
; 1421          1503  2 $logic_check (2, (IF .volb [volb$b_vol_format] EQL volb$k_vfmt_rt11 THEN (.ctx [ctx$l_cur_block] NEQ 0) ELSE
; 1422          1504  2
; 1423          1505  2 ! Get a pointer to the place to start scanning, and a pointer to the first byte past the end of the buffer
; 1424          1506  2 !
; 1425          1507  2 $logic_check (2, (.ctx [ctx$a_buffer] NEQ 0), 196);
; 1426          1508  2 buf_start = .ctx [ctx$a_buffer] + .ctx [ctx$l_cur_byte] +
; 1427          1509  2                 ((.ctx [ctx$l_cur_block] - .ctx [ctx$l_buf_base_block]) * 512);
; 1428          1510  2 buf_end = .ctx [ctx$a_buffer] +
; 1429          1511  2                 ((1 + .ctx [ctx$l_buf_high_block] - .ctx [ctx$l_buf_base_block]) * 512);
; 1430          1512  2
; 1431          1513  2 $$show_context;
; 1432          1514  2
; 1433          1515  2 ! Get the routine address for this specific record format
; 1434          1516  2 !
; 1435          1517  2 $trace_print_fao ('record format !UL', .filb [filb$b_rec_format]);
; 1436          1518  3 routn = (CASE .filb [filb$b_rec_format] FROM filb$k_rfmt_lobound TO filb$k_rfmt_hibound OF
; 1437          1519  3         SET
; 1438          1520  3             [filb$k_rfmt_binary] :      pdp_get_binary;
; 1439          1521  3             [filb$k_rfmt_fixed] :       pdp_get_fixed;
; 1440          1522  3             [filb$k_rfmt_stream] :      pdp_get_stream;
; 1441          1523  3             [INRANGE] :                 $exch_signal_return (exch$_invrecfmt);
; 1442          1524  3             [filb$k_rfmt_invalid,
; 1443          1525  3                 OUTRANGE] :             BEGIN $logic_check (0, (false), 243); 0 END;
; 1444          1526  2         TES);
; 1445          1527  2
; 1446          1528  2 ! Now call the routine and return the status from it
; 1447          1529  2 !
; 1448          1530  2 RETURN jsb_get (.routn, .filb, .buf_start, .buf_end);
; 1449          1531  2
; 1450          1532  1 END;


                                                                  .EXTRN  EXCH$_INVRECFMT

                                           07FC 00000             .ENTRY  EXCH$PDP_GET, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1451
                                                                          R10
                       5A 00000000G  EF  9E 00002             MOVAB   EXCH$UTIL_BLOCK_CHECK, R10                         ;
                       59 00000000G  00  9E 00009             MOVAB   LIB$STOP, R9                                       ;
                       58 00000000G  8F  D0 00010             MOVL    #EXCH$_BADLOGIC, R8                                ;
                       54        04  AC  D0 00017             MOVL    FILB, R4                                           ; 1493
                       52 035B00FA   8F  D0 0001B             MOVL    #56295674, R2                                      ; 1498
                       51      01C8  8F  3C 00022             MOVZWL  #456, R1                                           ;
                       50            54  D0 00027             MOVL    R4, R0                                             ;
                                     6A  16 0002A             JSB     EXCH$UTIL_BLOCK_CHECK                              ;
                       53        1C  A4  D0 0002C             MOVL    28(R4), R3                                         ; 1499
                       52 041B00F3   8F  D0 00030             MOVL    #68878579, R2                                      ;
                       51      01ED  8F  3C 00037             MOVZWL  #493, R1                                           ;
                       50            53  D0 0003C             MOVL    R3, R0                                             ;
                                     6A  16 0003F             JSB     EXCH$UTIL_BLOCK_CHECK                              ;
```

```
                                   7E     01EE   8F   3C 00041          MOVZWL  #494, -(SP)                          ; 1500
                                   52       20   A4   D0 00046          MOVL    32(R4), R2                           ;
                                                 52   DD 0004A          PUSHL   R2                                   ;
                         00000000G  00           02   FB 0004C          CALLS   #2, PDP_CHECK_CTX                    ;
                                   54       10   A2   D1 00053          CMPL    16(R2), R4                           ; 1501
                                                 0B   13 00057          BEQL    1$                                   ;
                                   7E       86   8F   9A 00059          MOVZBL  #134, -(SP)                          ;
                                                 01   DD 0005D          PUSHL   #1                                   ;
                                                 58   DD 0005F          PUSHL   R8                                   ;
                                   69            03   FB 00061          CALLS   #3, LIB$STOP                         ;
                                   53       14   A2   D1 00064 1$:      CMPL    20(R2), R3                           ; 1502
                                                 0B   13 00068          BEQL    2$                                   ;
                                   7E       87   8F   9A 0006A          MOVZBL  #135, -(SP)                          ;
                                                 01   DD 0006E          PUSHL   #1                                   ;
                                                 58   DD 00070          PUSHL   R8                                   ;
                                   69            03   FB 00072          CALLS   #3, LIB$STOP                         ;
                                   03       58   A3   91 00075 2$:      CMPB    88(R3), #3                           ; 1503
                                                 10   12 00079          BNEQ    3$                                   ;
                                            1C   A2   D5 0007B          TSTL    28(R2)                               ;
                                                 0B   12 0007E          BNEQ    3$                                   ;
                                   7E       B1   8F   9A 00080          MOVZBL  #177, -(SP)                          ;
                                                 01   DD 00084          PUSHL   #1                                   ;
                                                 58   DD 00086          PUSHL   R8                                   ;
                                   69            03   FB 00088          CALLS   #3, LIB$STOP                         ;
                                   53       18   A2   D0 0008B 3$:      MOVL    24(R2), R3                           ; 1507
                                                 0B   12 0008F          BNEQ    4$                                   ;
                                   7E       C4   8F   9A 00091          MOVZBL  #196, -(SP)                          ;
                                                 01   DD 00095          PUSHL   #1                                   ;
                                                 58   DD 00097          PUSHL   R8                                   ;
                                   69            03   FB 00099          CALLS   #3, LIB$STOP                         ;
                       51          53       24   A2   C1 0009C 4$:      ADDL3   36(R2), R3, R1                       ; 1508
                       50     1C   A2       2C   A2   C3 000A1          SUBL3   44(R2), 28(R2), R0                   ; 1509
                       50          50            09   78 000A7          ASHL    #9, R0, R0                           ;
                       56          51            50   C1 000AB          ADDL3   R0, R1, BUF_START                    ;
                       52     30   A2       2C   A2   C3 000AF          SUBL3   44(R2), 48(R2), R2                   ; 1511
                       52          52            09   78 000B5          ASHL    #9, R2, R2                           ;
                                   57     0200 C243   9E 000B9          MOVAB   512(R2)[R3], BUF_END                 ; 1510
                       03          00       28   A4   8F 000BF          CASEB   40(R4), #0, #3                       ; 1518
          0025        001E        0017          0008     000C4 5$:      .WORD   6$-5$,-                              ;
                                                                                7$-5$,-                              ;
                                                                                8$-5$,-                              ;
                                                                                9$-5$                                ;
                                   7E       F3   8F   9A 000CC 6$:      MOVZBL  #243, -(SP)                          ; 1525
                                                 01   DD 000D0          PUSHL   #1                                   ;
                                                 58   DD 000D2          PUSHL   R8                                   ;
                                   69            03   FB 000D4          CALLS   #3, LIB$STOP                         ;
                                                 50   D4 000D7          CLRL    ROUTN                                ;
                                                 13   11 000D9          BRB     10$                                  ;
                                   50    0000V   CF   9E 000DB 7$:      MOVAB   PDP_GET_BINARY, ROUTN                ; 1518
                                                 0C   11 000E0          BRB     10$                                  ;
                                   50    0000V   CF   9E 000E2 8$:      MOVAB   PDP_GET_FIXED, ROUTN                 ;
                                                 05   11 000E7          BRB     10$                                  ;
                                   50    0000V   CF   9E 000E9 9$:      MOVAB   PDP_GET_STREAM, ROUTN                ;
                                   55            54   D0 000EE 10$:     MOVL    R4, R5                               ; 1530
                                                 60   16 000F1          JSB     (ROUTN)                              ;
                                                      04 000F3          RET                                          ; 1532
```

```
; Routine Size:  244 bytes,    Routine Base:  EXCH$PDP_CODE + 068A
```

```
1 GLOBAL ROUTINE pdp_get_binary (filb : $ref_bblock,      %SBTTL 'pdp_get_binary (filb, buf_start, buf_end)'
1                                 buf_start, buf_end) : jsb_get =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Return a pointer to the next formatted binary record in the file
2 !
2 ! INPUTS:
2 !
2 !       filb      - pointer to filb for an open RT11 file
2 !       buf_start - pointer to next byte in the buffer
2 !       buf_end   - pointer to one past the end of buffer
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_get_binary> ');
2
2 LOCAL
2     new_start,                          ! Pointer to look next time.
2     tmp,
2     status
2     ;
2
2 BIND
2     ctx  = filb [filb$a_context]        : $ref_bblock,
2     volb = filb [filb$a_assoc_volb]     : $ref_bblock
2     ;
```

```
2 $debug_print_lit ('entry');
2
2 ! Attempt to find a record in the current portion of the buffer
2 !
2 status = pdp_find_binary_record (.filb, .buf_start, .buf_end, new_start);
2
2 ! What did we see, what do we do
2 !
2 CASE .status FROM findbin$k_lobound TO findbin$k_hibound OF
2 SET
2
2     ! Success, update our next record pointer and return true
2     !
2     [findbin$k_success, findbin$k_chksum] :
2
3                 BEGIN
3                 IF .status EQL findbin$k_chksum
3                 THEN
3                     $exch_signal (exch$_binchksum, 2, .filb [filb$l_result_name_len], filb [filb$t_result_na
3
3                 tmp = .new_start - .ctx [ctx$a_buffer]; ! Save the updated position for the next get
3                 ctx [ctx$l_cur_byte]  = .tmp MOD 512;
3                 ctx [ctx$l_cur_block] = (.tmp / 512) + .ctx [ctx$l_buf_base_block];
3                 RETURN true;                                    ! Found a record
2                 END;
2
2     ! Hit the end of the buffer with no record, determine if EOF or need to read more buffer
2     !
2     [findbin$k_eob] :
2
3                 BEGIN
3
3                 $trace_print_lit ('findbin$k_eob status');
3                 $$show_context;
3
3                 ! If we are already at the eof block, then we have found EOF and can return
3                 !
4                 IF (.ctx [ctx$l_buf_high_block] GEQU .ctx [ctx$l_eof_block])
4                   AND
4                    (.ctx [ctx$l_eof_block] NEQ -1)
3                 THEN
3                     status = false
3
3                 ! Otherwise, we can read in more data
3                 !
3                 ELSE
4                     BEGIN
5                     IF NOT (status = pdp_buffer_advance_read (.ctx))
4                     THEN
5                         BEGIN
5                         IF .status EQL exch$_stmrecfmt  ! Means no room to read more blocks
5                         THEN
6                             BEGIN
6                             status = exch$_binrecfmt;
6                             $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_na
6                             END
5                         ELSE
```

```
; 1558          1638  5                           RETURN .status;
; 1559          1639  5                       END
; 1560          1640  4                   ELSE
; 1561          1641  4                       RETURN exch$pdp_get (.filb);
; 1562          1642  3                   END;
; 1563          1643  3
; 1564          1644  2               END;
; 1565          1645  2
; 1566          1646  2       ! Found a badly formatted record
; 1567          1647  2       !
; 1568          1648  2       [findbin$k_bad_fmt] :
; 1569          1649  3               BEGIN
; 1570          1650  3               status = exch$_binrecfmt;
; 1571          1651  3               $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name]);
; 1572          1652  2               END;
; 1573          1653  2
; 1574          1654  2       [findbin$k_too_big] :
; 1575          1655  2
; 1576          1656  3               BEGIN
; 1577          1657  3               status = exch$_rectoobig;
; 1578          1658  3               $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name]);
; 1579          1659  2               END;
; 1580          1660  2
; 1581          1661  2       [INRANGE, OUTRANGE] :
; 1582          1662  2
; 1583          1663  2               $logic_check (0, (false), 244);
; 1584          1664  2
; 1585          1665  2   TES;
; 1586          1666  2
; 1587          1667  2   ! Set the next record position to invalid, and return the error
; 1588          1668  2   !
; 1589          1669  2   ctx [ctx$l_cur_byte]  = 0;
; 1590          1670  2   ctx [ctx$l_cur_block] = 0;
; 1591          1671  2
; 1592          1672  2   $$show_context;
; 1593          1673  2   $debug_print_lit ('returning status !XL', .status);
; 1594          1674  2
; 1595          1675  2   RETURN .status;
; 1596          1676  2
; 1597          1677  1   END;



                                                                .EXTRN  EXCH$_BINCHKSUM
                                                                .EXTRN  EXCH$_BINRECFMT

                         5E             04  C2 00000 PDP_GET_BINARY::
                                                                SUBL2   #4, SP                                          ; 1533
                                 40E0   8F  BB 00003            PUSHR   #^M<R5,R6,R7,SP>                                ; 1585
                   FD93   CF            04  FB 00007            CALLS   #4, PDP_FIND_BINARY_RECORD                      ;
                          53            50  D0 0000C            MOVL    R0, STATUS                                      ;
                   04     00            53  CF 0000F            CASEL   STATUS, #0, #4                                  ; 1589
     00A9          001F         0066  001F     00013 1$:        .WORD   2$-1$,-                                         ;
                                      00A0     0001B                    4$-1$,-                                         ;
                                                                        2$-1$,-                                         ;
                                                                        9$-1$,-                                         ;
                                                                        8$-1$                                           ;
```

```
                            7E         F4   8F  9A 0001D        MOVZBL  #244, -(SP)                           ; 1663
                                            01  DD 00021        PUSHL   #1                                    ;
                                 00000000G  8F  DD 00023        PUSHL   #EXCH$_BADLOGIC                       ;
                  00000000G  00             03  FB 00029        CALLS   #3, LIB$STOP                          ;
                                            5E  11 00030        BRB     5$                                    ;
                             02             53  D1 00032 2$:    CMPL    STATUS, #2                            ; 1597
                                            15  12 00035        BNEQ    3$                                    ;
                                       5A   A5  9F 00037        PUSHAB  90(FILB)                              ; 1599
                                       3A   A5  DD 0003A        PUSHL   58(FILB)                              ;
                                            02  DD 0003D        PUSHL   #2                                    ;
                                 00000000G  8F  DD 0003F        PUSHL   #EXCH$_BINCHKSUM                      ;
                  00000000G  00             04  FB 00045        CALLS   #4, LIB$SIGNAL                        ;
                             51        20   A5  D0 0004C 3$:    MOVL    32(FILB), R1                          ; 1601
           50                6E        18   A1  C3 00050        SUBL3   24(R1), NEW_START, TMP                ;
    7E     00                50             01  7A 00055        EMUL    #1, TMP, #0, -(SP)                    ; 1602
    52     52                8E  00000200   8F  7B 0005A        EDIV    #512, (SP)+, R2, R2                   ;
                        24   A1             52  D0 00063        MOVL    R2, 36(R1)                            ;
                             50  00000200   8F  C6 00067        DIVL2   #512, R0                              ; 1603
                        1C   A1      2C B140    9E 0006E        MOVAB   @44(R1)[R0], 28(R1)                   ;
                             50             01  D0 00074        MOVL    #1, R0                                ; 1604
                                            68  11 00077        BRB     13$                                   ;
                             50        20   A5  D0 00079 4$:    MOVL    32(FILB), R0                          ; 1618
                        20   A0        30   A0  D1 0007D        CMPL    48(R0), 32(R0)                        ;
                                            0E  1F 00082        BLSSU   6$                                    ;
                  FFFFFFFF   8F        20   A0  D1 00084        CMPL    32(R0), #-1                           ; 1620
                                            04  13 0008C        BEQL    6$                                    ;
                                            53  D4 0008E        CLRL    STATUS                                ; 1622
                                            42  11 00090 5$:    BRB     11$                                   ;
                                            50  DD 00092 6$:    PUSHL   R0                                    ; 1628
                      F7E9   CF             01  FB 00094        CALLS   #1, PDP_BUFFER_ADVANCE_READ           ;
                             53             50  D0 00099        MOVL    R0, STATUS                            ;
                             0B             53  E8 0009C        BLBS    STATUS, 7$                            ;
                  00000000G  8F             53  D1 0009F        CMPL    STATUS, #EXCH$_STMRECFMT              ; 1631
                                            0B  13 000A6        BEQL    8$                                    ;
                                            34  11 000A8        BRB     12$                                   ; 1638
                                            55  DD 000AA 7$:    PUSHL   FILB                                  ; 1641
                      FE5B   CF             01  FB 000AC        CALLS   #1, EXCH$PDP_GET                      ;
                                            2E  11 000B1        BRB     13$                                   ;
                             53  00000000G  8F  D0 000B3 8$:    MOVL    #EXCH$_BINRECFMT, STATUS              ; 1650
                                            07  11 000BA        BRB     10$                                   ; 1651
                             53  00000000G  8F  D0 000BC 9$:    MOVL    #EXCH$_RECTOOBIG, STATUS              ; 1657
                                       5A   A5  9F 000C3 10$:   PUSHAB  90(FILB)                              ; 1658
                                       3A   A5  DD 000C6        PUSHL   58(FILB)                              ;
                                            02  DD 000C9        PUSHL   #2                                    ;
                                            53  DD 000CB        PUSHL   STATUS                                ;
                  00000000G  00             04  FB 000CD        CALLS   #4, LIB$SIGNAL                        ;
                             50        20   A5  D0 000D4 11$:   MOVL    32(FILB), R0                          ; 1669
                                       24   A0  D4 000D8        CLRL    36(R0)                                ;
                                       1C   A0  D4 000DB        CLRL    28(R0)                                ; 1670
                             50             53  D0 000DE 12$:   MOVL    STATUS, R0                            ; 1675
                             5E             04  C0 000E1 13$:   ADDL2   #4, SP                                ; 1677
                                            05 000E4            RSB                                           ;

; Routine Size:  229 bytes,    Routine Base:  EXCH$PDP_CODE + 077E
```

```
1 GLOBAL ROUTINE pdp_get_fixed (filb : $ref_bblock,       %SBTTL 'pdp_get_fixed (filb, buf_start, buf_end)'
1                                 buf_start, buf_end) : jsb_get =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Return a pointer to the next fixed-length record in the file
2 !
2 ! INPUTS:
2 !
2 !       filb      - pointer to filb for an open RT11 file
2 !       buf_start - pointer to next byte in the buffer
2 !       buf_end   - pointer to one past the end of buffer
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_get_fixed> ');
2
2 REGISTER
2     five12,
2     rec_size
2     ;
2
2 LOCAL
2     new_start,                          ! Pointer to look next time.
2     tmp,
2     status
2     ;
2
2 BIND
2     ctx  = filb [filb$a_context]        : $ref_bblock,
2     volb = filb [filb$a_assoc_volb]     : $ref_bblock
2     ;
2
2 $debug_print_lit ('entry');
2
2 ! Preset some registers for a bit more speed
```

```
2 !
2 five12 = 512;
2 rec_size = .filb [filb$l_fixed_len];
2
2 ! Get a pointer to the start of the next record
2 !
2 new_start = .buf_start + .rec_size;
2
2 ! See if the next record is in the buffer, EOF or advance the buffer if it isn't
2 !
2 IF (.new_start - 1) GEQU .buf_end
2 THEN
3     BEGIN
3
3     ! If the EOF block is in the buffer
3     !
4     IF (.ctx [ctx$l_buf_high_block] GEQU .ctx [ctx$l_eof_block])
3       AND
4        (.ctx [ctx$l_eof_block] NEQ -1)
3     THEN
4         BEGIN
4
4         ! Set the next record position to invalid, and return false
4         !
4         ctx [ctx$l_cur_byte]  = 0;
4         ctx [ctx$l_cur_block] = 0;
4         RETURN false;
4         END
4
4     ! Otherwise, read some more data in and recursively retry the get
4     !
3     ELSE
4         BEGIN
5         IF NOT (status = pdp_buffer_advance_read (.ctx))
4         THEN
4             RETURN .status;
4         RETURN exch$pdp_get (.filb);            ! And then try it again
3         END;
2     END;
2
2 $logic_check (2, ((.new_start - 1) LSSU .buf_end), 133);
2
2 ! Use locate mode - point the filb record info at the buffer
2 !
2 filb [filb$a_record] = .buf_start;
2 filb [filb$l_record_len] = .rec_size;
2
2 ! Update the next record position
2 !
2 $logic_check (2, (.ctx [ctx$a_buffer] NEQ 0), 198);
2 tmp = .new_start - .ctx [ctx$a_buffer]; ! Save the updated position for the next get
2 ctx [ctx$l_cur_byte]  = .tmp MOD .five12;
2 ctx [ctx$l_cur_block] = (.tmp / .five12) + .ctx [ctx$l_buf_base_block];
2
2 RETURN true;                                        ! Found a record
2
1 END;
```

```
                                   52      0200  8F  3C 00000 PDP_GET_FIXED::
                                                                      MOVZWL  #512, FIVE12                              ; 1736
                                   53        35  A5  D0 00005             MOVL    53(FILB), REC_SIZE                        ; 1737
                    54             56            53  C1 00009             ADDL3   REC_SIZE, BUF_START, NEW_START            ; 1741
                                             FF  A4  9F 0000D             PUSHAB  -1(R4)                                    ; 1745
                                   57            6E  D1 00010             CMPL    (SP), BUF_END                             ;
                                                 32  1F 00013             BLSSU   2$                                        ;
                                   50        20  A5  D0 00015             MOVL    32(FILB), R0                              ; 1751
                           20      A0        30  A0  D1 00019             CMPL    48(R0), 32(R0)                            ;
                                                 14  1F 0001E             BLSSU   1$                                        ;
                     FFFFFFFF      8F        20  A0  D1 00020             CMPL    32(R0), #-1                               ; 1753
                                                 0A  13 00C28             BEQL    1$                                        ;
                                             24  A0  D4 0002A             CLRL    36(R0)                                    ; 1759
                                             1C  A0  D4 0002D             CLRL    28(R0)                                    ; 1760
                                                 50  D4 00030             CLRL    R0                                        ; 1767
                                                 56  11 00032             BRB     4$                                        ;
                                                 50  DD 00034 1$:         PUSHL   R0                                        ; 1768
                         F762      CF            01  FB 00036             CALLS   #1, PDP_BUFFER_ADVANCE_READ               ;
                                   4C            50  E9 0003B             BLBC    STATUS, 4$                                ;
                                                 55  DD 0003E             PUSHL   FILB                                      ; 1771
                         FDE2      CF            01  FB 00040             CALLS   #1, EXCH$PDP_GET                          ;
                                                 43  11 00045             BRB     4$                                        ; 1767
                           46      A5            56  D0 00047 2$:         MOVL    BUF_START, 70(FILB)                       ; 1779
                           42      A5            53  D0 0004B             MOVL    REC_SIZE, 66(FILB)                        ; 1780
                                   53        20  A5  D0 0004F             MOVL    32(FILB), R3                              ; 1784
                                             18  A3  D5 00053             TSTL    24(R3)                                    ;
                                                 13  12 00056             BNEQ    3$                                        ;
                                   7E        C6  8F  9A 00058             MOVZBL  #198, -(SP)                               ;
                                                 01  DD 0005C             PUSHL   #1                                        ;
                                      00000000G  8F  DD 0005E             PUSHL   #EXCH$_BADLOGIC                           ;
                    00000000G      00            03  FB 00064             CALLS   #3, LIB$STOP                              ;
                    50             54        18  A3  C3 0006B 3$:         SUBL3   24(R3), NEW_START, TMP                    ; 1785
      7E            00             50            01  7A 00070             EMUL    #1, TMP, #0, -(SP)                        ; 1786
      51            51             8E            52  7B 00075             EDIV    FIVE12, (SP)+, R1, R1                     ;
                           24      A3            51  D0 0007A             MOVL    R1, 36(R3)                                ;
                                   50            52  C6 0007E             DIVL2   FIVE12, R0                                ; 1787
                           1C      A3       2C B340  9E 00081             MOVAB   @44(R3)[R0], 28(R3)                       ;
                                   50            01  D0 00087             MOVL    #1, R0                                    ; 1789
                                   5E            04  C0 0008A 4$:         ADDL2   #4, SP                                    ; 1791
                                                     05 0008D             RSB                                               ;

; Routine Size:  142 bytes,    Routine Base:  EXCH$PDP_CODE + 0863
```

```
1 GLOBAL ROUTINE pdp_get_stream (filb : $ref_bblock,      %SBTTL 'pdp_get_stream (filb, buf_start, buf_end)'
1                                buf_start, buf_end) : jsb_get =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Return a pointer to the next stream record in the file
2 !
2 ! INPUTS:
2 !
2 !       filb      - pointer to filb for an open RT11 file
2 !       buf_start - pointer to next byte in the buffer
2 !       buf_end   - pointer to one past the end of buffer
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_get_stream> ');
2
2 LOCAL
2     new_start,                          ! Pointer to look next time.
2     find_stat,
2     status
2     ;
2
2 BIND
2     ctx  = filb [filb$a_context]        : $ref_bblock,
2     volb = filb [filb$a_assoc_volb]     : $ref_bblock
2     ;
```

```
2 $debug_print_lit ('entry');
2
2 ! Attempt to find a record in this portion of the buffer
2 !
2 find_stat = pdp_find_stream_record (.filb, .buf_start, .buf_end, new_start);
2
2 ! What did we see, what do we do
2 !
2 CASE .find_stat FROM findstm$k_lobound TO findstm$k_hibound OF
2 SET
2
2     ! Success, update our next record pointer and return true
2     !
2     [findstm$k_success] :
2
3                 BEGIN
3                 LOCAL
3                     tmp;
3                 tmp = .new_start - .ctx [ctx$a_buffer]; ! Save the updated position for the next get
3                 ctx [ctx$l_cur_byte]  = .tmp MOD 512;
3                 ctx [ctx$l_cur_block] = (.tmp / 512) + .ctx [ctx$l_buf_base_block];
3                 RETURN true;                                 ! Found a record
2                 END;
2
2     ! Found a control Z at the start of a record, done with this file
2     !
2     [findstm$k_ctrlz_eof] :
2
2                 status = false;
2
2     ! Hit the end of the buffer with no record, determine if EOF or need to read more buffer
2     !
2     [findstm$k_eob] :
2
3                 BEGIN
3
3                 $trace_print_lit ('findstm$k_eob status');
3                 $$show_context;
3
3                 ! If we are already at the eof block, then we have found EOF and can return
3                 !
4                 IF (.ctx [ctx$l_buf_high_block] GEQU .ctx [ctx$l_eof_block])
3                   AND
4                   (.ctx [ctx$l_eof_block] NEQ -1)
3                 THEN
3                     status = false
3
3                 ! Otherwise, we can read in more data
3                 !
3                 ELSE
4                     BEGIN
5                     IF NOT (status = pdp_buffer_advance_read (.ctx))
4                     THEN
5                         BEGIN
5                         IF .status EQL exch$_stmrecfmt  ! Means no room to read more blocks
5                         THEN
5                             $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_na
5                                      ELSE
5                                          RETURN .status;
5                                      END
4                                  ELSE
4                                      RETURN exch$pdp_get (.filb);
3                                  END;
3
2                          END;
2
2          ! Hit the end of the buffer with some record, determine if can read more buffer or final record is missi
2          !
2          [findstm$k_no_term] :
2
3                          BEGIN
3
3                          $trace_print_lit ('findstm$k_no_term status');
3                          $$show_context;
3
3                          ! If we are already at the eof block, then the record reaches to the end of the block
3                          !
4                          IF (.ctx [ctx$l_buf_high_block] GEQU .ctx [ctx$l_eof_block])
3                            AND
4                             (.ctx [ctx$l_eof_block] NEQ -1)
3                          THEN
4                              BEGIN
4                              LOCAL
4                                  tmp;
4                              tmp = .new_start - .ctx [ctx$a_buffer];      ! Save the updated position for the next get
4                              ctx [ctx$l_cur_byte]  = .tmp MOD 512;
4                              ctx [ctx$l_cur_block] = (.tmp / 512) + .ctx [ctx$l_buf_base_block];
4                              RETURN true;                                        ! Found a record
4                              END
4
4                          ! Otherwise, we can read in more data
4                          !
3                          ELSE
4                              BEGIN
5                              IF NOT (status = pdp_buffer_advance_read (.ctx))
4                              THEN
5                                  BEGIN
5                                  IF .status EQL exch$_stmrecfmt  ! Means no room to read more blocks
5                                  THEN
5                                      $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_na
5                                  ELSE
5                                      RETURN .status;
5                                  END
4                              ELSE
4                                  RETURN exch$pdp_get (.filb);
3                              END;
3
2                          END;
2
2          ! Found a badly formatted record
2          !
2          [findstm$k_bad_fmt] :
2
3                          BEGIN
```

```
; 1877          1954  3                  status = exch$_stmrecfmt;
; 1878          1955  3                  $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name]);
; 1879          1956  2                  END;
; 1880          1957  2
; 1881          1958  2      [INRANGE, OUTRANGE] :
; 1882          1959  2
; 1883          1960  2                  $logic_check (0, (false), 245);
; 1884          1961  2
; 1885          1962  2  TES;
; 1886          1963  2
; 1887          1964  2  ! Set the next record position to invalid, and return false
; 1888          1965  2  !
; 1889          1966  2  ctx [ctx$l_cur_byte]  = 0;
; 1890          1967  2  ctx [ctx$l_cur_block] = 0;
; 1891          1968  2
; 1892          1969  2  RETURN .status;
; 1893          1970  2
; 1894          1971  1  END;



                               5E                 04  C2 00000 PDP_GET_STREAM::
                                                               SUBL2   #4, SP                                  ; 1792
                                          40E0    8F  BB 00003         PUSHR   #^M<R5,R6,R7,SP>                ; 1844
                          FCCF   CF               04  FB 00007         CALLS   #4, PDP_FIND_STREAM_RECORD
                    04           00               50  CF 0000C         CASEL   FIND_STAT, #0, #4               ; 1848
       0042       0025         003A             001F     00010 1$:     .WORD   2$-1$,-                         ;
                                                00A1     00018                 4$-1$,-                         ;
                                                                               3$-1$,-                         ;
                                                                               7$-1$,-                         ;
                                                                               12$-1$                          ;
                               7E         F5      8F  9A 0001A         MOVZBL  #245, -(SP)                     ; 1960
                                                  01  DD 0001E         PUSHL   #1                              ;
                                     00000000G    8F  DD 00020         PUSHL   #EXCH$_BADLOGIC                 ;
                      00000000G  00               03  FB 00026         CALLS   #3, LIB$STOP                    ;
                                                  1D  11 0002D         BRB     5$                              ;
                               51         20      A5  D0 0002F 2$:     MOVL    32(FILB), R1                    ; 1858
                                                  32  11 00033         BRB     8$                              ;
                               50         20      A5  D0 00035 3$:     MOVL    32(FILB), R0                    ; 1881
                          20   A0         30      A0  D1 00039         CMPL    48(R0), 32(R0)                  ;
                                                  0E  1F 0003E         BLSSU   6$                              ;
                      FFFFFFFF 8F         20      A0  D1 00040         CMPL    32(R0), #-1                     ; 1883
                                                  04  13 00048         BEQL    6$                              ;
                                                  53  D4 0004A 4$:     CLRL    STATUS                          ; 1885
                                                  7B  11 0004C 5$:     BRB     14$                             ;
                                                  50  DD 0004E 6$:     PUSHL   R0                              ; 1891
                                                  40  11 00050         BRB     10$                             ;
                               51         20      A5  D0 00052 7$:     MOVL    32(FILB), R1                    ; 1917
                          20   A1         30      A1  D1 00056         CMPL    48(R1), 32(R1)                  ;
                                                  33  1F 0005B         BLSSU   9$                              ;
                      FFFFFFFF 8F         20      A1  D1 0005D         CMPL    32(R1), #-1                     ; 1919
                                                  29  13 00065         BEQL    9$                              ;
                    50         6E         18      A1  C3 00067 8$:     SUBL3   24(R1), NEW_START, TMP          ; 1924
         7E         00         50                 01  7A 0006C         EMUL    #1, TMP, #0, -(SP)              ; 1925
         52         52         8E  00000200       8F  7B 00071         EDIV    #512, (SP)+, R2, R2             ;
```

```
              24  A1            52  D0 0007A          MOVL     R2, 36(R1)                              ;
                  50 00000200   8F  C6 0007E          DIVL2    #512, R0                                ; 1926
              1C  A1       2C B140  9E 00085          MOVAB    @44(R1)[R0], 28(R1)                     ;
                  50            01  D0 0008B          MOVL     #1, R0                                  ; 1927
                                46  11 0008E          BRB      16$                                     ;
                                51  DD 00090 9$:      PUSHL    R1                                      ; 1934
            F678  CF            01  FB 00092 10$:     CALLS    #1, PDP_BUFFER_ADVANCE_READ             ;
                  53            50  D0 00097          MOVL     R0, STATUS                              ;
                  0B            53  E8 0009A          BLBS     STATUS, 11$                             ;
       00000000G  8F            53  D1 0009D          CMPL     STATUS, #EXCH$_STMRECFMT                ; 1937
                                12  13 000A4          BEQL     13$                                     ;
                                2B  11 000A6          BRB      15$                                     ; 1941
                                55  DD 000A8 11$:     PUSHL    FILB                                    ; 1944
            FCEA  CF            01  FB 000AA          CALLS    #1, EXCH$PDP_GET                        ;
                                25  11 000AF          BRB      16$                                     ;
                  53 00000000G  8F  D0 000B1 12$:     MOVL     #EXCH$_STMRECFMT, STATUS                ; 1954
                           5A   A5  9F 000B8 13$:     PUSHAB   90(FILB)                                ; 1955
                           3A   A5  DD 000BB          PUSHL    58(FILB)                                ;
                                02  DD 000BE          PUSHL    #2                                      ;
                                53  DD 000C0          PUSHL    STATUS                                  ;
       00000000G  00            04  FB 000C2          CALLS    #4, LIB$SIGNAL                          ;
                  50       20   A5  D0 000C9 14$:     MOVL     32(FILB), R0                            ; 1966
                           24   A0  D4 000CD          CLRL     36(R0)                                  ;
                           1C   A0  D4 000D0          CLRL     28(R0)                                  ; 1967
                  50            53  D0 000D3 15$:     MOVL     STATUS, R0                              ; 1969
                  5E            04  C0 000D6 16$:     ADDL2    #4, SP                                  ; 1971
                                05  000D9             RSB                                              ;

; Routine Size:  218 bytes,    Routine Base:  EXCH$PDP_CODE + 08F1
```

```
1 GLOBAL ROUTINE exch$pdp_put =   %SBTTL 'exch$pdp_put'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Common dispatch for RT11-style put routines.  The main purpose of the extra dispatch is simplify the
2 !       mechanism for optimizing i/o transfers to physical mode when possible (for example RT11 -> RT11 does
2 !       need record mode).
2 !
2 ! INPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       see the BIND expression
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       see the BIND expression
2 !
2 ! ROUTINE VALUE:
2 !
2 !       value of format-specific put routine
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('pdp_put> ');
2
2 LOCAL
2     buf_start,
2     buf_end,
2     routn
2     ;
2
2 BIND
2     copy = exch$a_gbl [excg$a_copy_work]: $ref_bblock,  ! COPY verb work area
2     inp_filb = copy [copy$a_inp_filb]   : $ref_bblock,  ! pointer to the input filb with the record info
2     out_filb = copy [copy$a_out_filb]   : $ref_bblock,  ! pointer to filb for an open Files-11 output file
2     len  = inp_filb [filb$l_record_len],                ! length of the record
2     buf  = inp_filb [filb$a_record],                    ! address of the record
2     ctx  = out_filb [filb$a_context]    : $ref_bblock,  ! output file context block
2     volb = out_filb [filb$a_assoc_volb] : $ref_bblock   ! output file volume block
2     ;
2
2 $debug_print_fao ('entry, format=!UL, len=!UL, buf[0:19]="!AF"', .out_filb [filb$b_rec_format], .len, 20, .b
2
2 $block_check (2, .inp_filb, filb, 466);
2 $block_check (2, .out_filb, filb, 467);
```

```
; 1953      2029  2 $check_call (1, pdp_check_ctx, .ctx, 537);             ! $block_check (1, .ctx, (dos11ctx or rt11ctx), 537)
; 1954      2030  2 $block_check (2, .volb, volb, 468);
; 1955      2031  2 $logic_check (2, (.ctx [ctx$a_assoc_filb] EQL .out_filb), 168);
; 1956      2032  2 $logic_check (2, (.ctx [ctx$a_assoc_volb] EQL .volb), 169);
; 1957      2033  2 $logic_check (2, (IF .volb [volb$b_vol_format] EQL volb$k_vfmt_rt11 THEN (.ctx [ctx$l_cur_block] NEQ 0) ELSE
; 1958      2034  2 $logic_check (2, (.len LEQU filb$s_record_buffer), 283);
; 1959      2035  2
; 1960      2036  2 ! Get pointers to the start of the next record position in the buffer, and to the end of the current buffer
; 1961      2037  2 !
; 1962      2038  2 $logic_check (2, (.ctx [ctx$a_buffer] NEQ 0), 200);
; 1963      2039  2 buf_start = .ctx [ctx$a_buffer] + .ctx [ctx$l_cur_byte] +
; 1964      2040  2             ((.ctx [ctx$l_cur_block] - .ctx [ctx$l_buf_base_block]) * 512);
; 1965      2041  2 buf_end   = .ctx [ctx$a_buffer] +
; 1966      2042  2             ((1 + .ctx [ctx$l_buf_high_block] - .ctx [ctx$l_buf_base_block]) * 512);
; 1967      2043  2
; 1968      2044  2 ! Get the address of the record format specific routine
; 1969      2045  2 !
; 1970      2046  2 $trace_print_fao ('record format !UL', .out_filb [filb$b_rec_format]);
; 1971      2047  3 routn = (CASE .out_filb [filb$b_rec_format] FROM filb$k_rfmt_lobound TO filb$k_rfmt_hibound OF
; 1972      2048  3         SET
; 1973      2049  3             [filb$k_rfmt_binary] :      pdp_put_binary;
; 1974      2050  3             [filb$k_rfmt_fixed] :       pdp_put_fixed;
; 1975      2051  3             [filb$k_rfmt_stream] :      pdp_put_stream;
; 1976      2052  3             [INRANGE] :                 $exch_signal_return (exch$_invrecfmt);
; 1977      2053  3             [filb$k_rfmt_invalid,
; 1978      2054  3                 OUTRANGE] :             BEGIN $logic_check (0, (false), 246); 0 END;
; 1979      2055  2         TES);
; 1980      2056  2
; 1981      2057  2 ! Now call that routine, returning the value of the routine
; 1982      2058  2 !
; 1983      2059  2 RETURN jsb_put (.routn, .buf_start, .buf_end, .ctx, .len, .buf);
; 1984      2060  1 END;



                                                               .EXTRN  EXCH$A_GBL

                                          0FFC 00000           .ENTRY  EXCH$PDP_PUT, Save R2,R3,R4,R5,R6,R7,R8,R9,-;  1972
                                                                       R10,R11                                      ;
                       5B 00000000G  8F   D0 00002             MOVL    #EXCH$_BADLOGIC, R11                         ;
          50 00000000G  EF           04   C1 00009             ADDL3   #4, EXCH$A_GBL, R0                           ;  2016
          53            60           3C   C1 00011             ADDL3   #60, (R0), R3                                ;  2017
          50            60 00000044  8F   C1 00015             ADDL3   #68, (R0), R0                                ;  2018
          56            63 00000042  8F   C1 0001D             ADDL3   #66, (R3), R6                                ;  2019
          57            63 00000046  8F   C1 00025             ADDL3   #70, (R3), R7                                ;  2020
                        55           60   D0 0002D             MOVL    (R0), R5                                     ;  2021
                        52 035B00FA  8F   D0 00030             MOVL    #56295674, R2                                ;  2027
                        51     01D2  8F   3C 00037             MOVZWL  #466, R1                                     ;
                        50           63   D0 0003C             MOVL    (R3), R0                                     ;
                           00000000G EF   16 0003F             JSB     EXCH$UTIL_BLOCK_CHECK                        ;
                        52 035B00FA  8F   D0 00045             MOVL    #56295674, R2                                ;  2028
                        51     01D3  8F   3C 0004C             MOVZWL  #467, R1                                     ;
                        50           55   D0 00051             MOVL    R5, R0                                       ;
                           00000000G EF   16 00054             JSB     EXCH$UTIL_BLOCK_CHECK                        ;
                        7E     0219  8F   3C 0005A             MOVZWL  #537, -(SP)                                  ;  2029
                        53       20  A5   D0 0005F             MOVL    32(R5), R3                                   ;
                                     53   DD 00063             PUSHL   R3                                           ;
```

```
                    00000000G  00            02  FB 00065      CALLS   #2, PDP_CHECK_CTX                 ;
                               54       1C   A5  D0 0006C      MOVL    28(R5), R4                        ; 2030
                               52 041B00F3   8F  D0 00070      MOVL    #68878579, R2                     ;
                               51     01D4   8F  3C 00077      MOVZWL  #468, R1                          ;
                               50            54  D0 0007C      MOVL    R4, R0                            ;
                                  00000000G  EF  16 0007F      JSB     EXCH$UTIL_BLOCK_CHECK             ;
                               55       10   A3  D1 00085      CMPL    16(R3), R5                        ; 2031
                                             0F  13 00089      BEQL    1$                                ;
                               7E       A8   8F  9A 0008B      MOVZBL  #168, -(SP)                       ;
                                             01  DD 0008F      PUSHL   #1                                ;
                                             5B  DD 00091      PUSHL   R11                               ;
                    00000000G  00            03  FB 00093      CALLS   #3, LIB$STOP                      ;
                               54       14   A3  D1 0009A 1$:  CMPL    20(R3), R4                        ; 2032
                                             0F  13 0009E      BEQL    2$                                ;
                               7E       A9   8F  9A 000A0      MOVZBL  #169, -(SP)                       ;
                                             01  DD 000A4      PUSHL   #1                                ;
                                             5B  DD 000A6      PUSHL   R11                               ;
                    00000000G  00            03  FB 000A8      CALLS   #3, LIB$STOP                      ;
                               03       58   A4  91 000AF 2$:  CMPB    88(R4), #3                        ; 2033
                                             14  12 000B3      BNEQ    3$                                ;
                                        1C   A3  D5 000B5      TSTL    28(R3)                            ;
                                             0F  12 000B8      BNEQ    3$                                ;
                               7E       B0   8F  9A 000BA      MOVZBL  #176, -(SP)                       ;
                                             01  DD 000BE      PUSHL   #1                                ;
                                             5B  DD 000C0      PUSHL   R11                               ;
                    00000000G  00            03  FB 000C2      CALLS   #3, LIB$STOP                      ;
                    00000200   8F            66  D1 000C9 3$:  CMPL    (R6), #512                        ; 2034
                                             10  1B 000D0      BLEQU   4$                                ;
                               7E     011B   8F  3C 000D2      MOVZWL  #283, -(SP)                       ;
                                             01  DD 000D7      PUSHL   #1                                ;
                                             5B  DD 000D9      PUSHL   R11                               ;
                    00000000G  00            03  FB 000DB      CALLS   #3, LIB$STOP                      ;
                               52       18   A3  D0 000E2 4$:  MOVL    24(R3), R2                        ; 2038
                                             0F  12 000E6      BNEQ    5$                                ;
                               7E       C8   8F  9A 000E8      MOVZBL  #200, -(SP)                       ;
                                             01  DD 000EC      PUSHL   #1                                ;
                                             5B  DD 000EE      PUSHL   R11                               ;
                    00000000G  00            03  FB 000F0      CALLS   #3, LIB$STOP                      ;
                51             52       24   A3  C1 000F7 5$:  ADDL3   36(R3), R2, R1                    ; 2039
                50        1C   A3       2C   A3  C3 000FC      SUBL3   44(R3), 28(R3), R0                ; 2040
                50             50            09  78 00102      ASHL    #9, R0, R0                        ;
                59             51            50  C1 00106      ADDL3   R0, R1, BUF_START                 ;
                50        30   A3       2C   A3  C3 0010A      SUBL3   44(R3), 48(R3), R0                ; 2042
                50             50            09  78 00110      ASHL    #9, R0, R0                        ;
                               5A     0200 C042  9E 00114      MOVAB   512(R0)[R2], BUF_END              ; 2041
                03             00       28   A5  8F 0011A      CASEB   40(R5), #0, #3                    ; 2047
      0029    0022           001B          0008     0011F 6$:  .WORD   7$-6$,-                           ;
                                                                       8$-6$,-                           ;
                                                                       9$-6$,-                           ;
                                                                       10$-6$                            ;
                               7E       F6   8F  9A 00127 7$:  MOVZBL  #246, -(SP)                       ; 2054
                                             01  DD 0012B      PUSHL   #1                                ;
                                             5B  DD 0012D      PUSHL   R11                               ;
                    00000000G  00            03  FB 0012F      CALLS   #3, LIB$STOP                      ;
                                             50  D4 00136      CLRL    ROUTN                             ;
                                             13  11 00138      BRB     11$                               ;
                               50    0000V   CF  9E 0013A 8$:  MOVAB   PDP_PUT_BINARY, ROUTN             ; 2047
```

```
                                    0C  11 0013F           BRB     11$                                  :
                      50    0000V   CF  9E 00141 9$:       MOVAB   PDP_PUT_FIXED, ROUTN                 :
                                    05  11 00146           BRB     11$                                  :
                      50    0000V   CF  9E 00148 10$:      MOVAB   PDP_PUT_STREAM, ROUTN                :
                                    67  DD 0014D 11$:      PUSHL   (R7)                                 : 2059
                                    66  DD 0014F           PUSHL   (R6)                                 :
                                    53  DD 00151           PUSHL   R3                                   :
                                    60  16 00153           JSB     (ROUTN)                              :
                                        04 00155           RET                                          : 2060

; Routine Size:  342 bytes,    Routine Base:  EXCH$PDP_CODE + 09CB
```

```
GLOBAL ROUTINE pdp_put_binary (buf_start, buf_end, ctx : $ref_bblock, len, buf) : jsb_put =     %SBTTL 'pdp_
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       Add the next formatted binary record in the file
!
! INPUTS:
!
!       buf_start - Pointer to next byte in the buffer
!       buf_end   - Pointer to one past the end of buffer
!       ctx       - Output file context block
!       len       - Length of the record to be put
!       buf       - Address of the record
!
! IMPLICIT INPUTS:
!
!       see the BIND expression
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       see the BIND expression
!
! ROUTINE VALUE:
!
!       true if success, false if any error
!
! SIDE EFFECTS:
!
!       error conditions will be signaled
!--

$dbgtrc_prefix ('pdp_put_binary> ');

REGISTER
    next_rec,
    tmp
    ;

BIND
    copy = exch$a_gbl [excg$a_copy_work]: $ref_bblock,  ! COPY verb work area
    out_filb = copy [copy$a_out_filb]   : $ref_bblock   ! pointer to filb for an open Files-11 output file
    ;

$debug_print_fao ('entry, len=!UL, buf[0:19]="!AF"', .len, 20, .buf);

! Get a pointer to the start of the next record after this one
!
next_rec = .buf_start + .len + 5;               ! <sentinel-word> <length-word> <record-data> <checksum-byte

! See if the next record will fit in the buffer, EOF or advance the buffer if it isn't
!
```

```
; 2043          2118   2 IF (.next_rec - 1) GEQU .buf_end
; 2044          2119   2 THEN
; 2045          2120   2     RETURN pdp_buffer_check (.ctx, .out_filb);
; 2046          2121   2
; 2047          2122   2 ! Move the record to the buffer
; 2048          2123   2 !
; 2049          2124   2 pdp_copy_binary_record (.len, .buf, .buf_start);
; 2050          2125   2
; 2051          2126   2 ! Update the next record position and return
; 2052          2127   2 !
; 2053          2128   2 RETURN pdp_buffer_update (.ctx, .next_rec);
; 2054          2129   2
; 2055          2130   1 END;



                            55              59  D0 00000 PDP_PUT_BINARY::
                                                                 MOVL    R9, R5                                        ; 2061
                  50 00000000G  EF          04  C1 00003         ADDL3   #4, EXCH$A_GBL, R0                            ; 2106
                  50            60 00000044 8F  C1 0000B         ADDL3   #68, (R0), R0                                 ; 2107
                  59            55       08 AE  C1 00013         ADDL3   LEN, BUF_START, R9                            ; 2114
                                54       05 A9  9E 00018         MOVAB   5(R9), NEXT_REC                               ;
                                59       FF A4  9E 0001C         MOVAB   -1(R4), R9                                    ; 2118
                                5A          59  D1 00020         CMPL    R9, BUF_END                                   ;
                                            0A  1F 00023         BLSSU   1$                                            ;
                                53          60  D0 00025         MOVL    (R0), R3                                      ; 2120
                                52       04 AE  D0 00028         MOVL    CTX, R2                                       ;
                                          F7B3  31 0002C         BRW     PDP_BUFFER_CHECK                              ;
                                            55  DD 0002F 1$:     PUSHL   BUF_START                                     ; 2124
                                         10 AE  DD 00031         PUSHL   BUF                                           ;
                                         10 AE  DD 00034         PUSHL   LEN                                           ;
                          F902  CF          03  FB 00037         CALLS   #3, PDP_COPY_BINARY_RECORD                    ;
                                53          54  D0 0003C         MOVL    NEXT_REC, R3                                  ; 2128
                                52       04 AE  D0 0003F         MOVL    CTX, R2                                       ;
                                          F7CF  31 00043         BRW     PDP_BUFFER_UPDATE                             ;

; Routine Size:  70 bytes,    Routine Base:  EXCH$PDP_CODE + 0B21
```

```
GLOBAL ROUTINE pdp_put_fixed (buf_start, buf_end, ctx : $ref_bblock, len, buf) : jsb_put =      %SBTTL 'pdp_
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
! INPUTS:
!
!       buf_start - Pointer to next byte in the buffer
!       buf_end   - Pointer to one past the end of buffer
!       ctx       - Output file context block
!       len       - Length of the record to be put
!       buf       - Address of the record
!
! IMPLICIT INPUTS:
!
!       see the BIND expression
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       see the BIND expression
!
! ROUTINE VALUE:
!
!       true if success, false if any error
!
! SIDE EFFECTS:
!
!       error conditions will be signaled
!--

$dbgtrc_prefix ('pdp_put_fixed> ');

REGISTER
    rec_size,
    next_rec,                           ! Pointer to look next time.
    tmp
    ;

BIND
    copy = exch$a_gbl [excg$a_copy_work]: $ref_bblock,  ! COPY verb work area
    out_filb = copy [copy$a_out_filb]   : $ref_bblock   ! pointer to filb for an open Files-11 output file
    ;

$debug_print_fao ('entry, len=!UL, buf[0:19]="!AF"', .len, 20, .buf);

rec_size = .out_filb [filb$l_fixed_len];

! Get a pointer to the start of the next record after this one
!
next_rec = .buf_start + .rec_size;

! See if the next record will fit in the buffer, EOF or advance the buffer if it isn't
```

```
; 2114          2188  2 !
; 2115          2189  2 IF (.next_rec - 1) GEQU .buf_end
; 2116          2190  2 THEN
; 2117          2191  2     RETURN pdp_buffer_check (.ctx, .out_filb);
; 2118          2192  2
; 2119          2193  2 ! Move the record to the buffer
; 2120          2194  2 !
; 2121          2195  2 CH$COPY (.len, .buf, .out_filb [filb$b_pad_char], .rec_size, .buf_start);
; 2122          2196  2
; 2123          2197  2 ! Update the next record position and return
; 2124          2198  2 !
; 2125          2199  2 RETURN pdp_buffer_update (.ctx, .next_rec);
; 2126          2200  2
; 2127          2201  1 END;



                  50 00000000G  EF          04  C1 00000 PDP_PUT_FIXED::
                                                             ADDL3   #4, EXCH$A_GBL, R0                             ; 2175
                  50            60 00000044 8F  C1 00008     ADDL3   #68, (R0), R0                                  ; 2176
                                55          60  D0 00010     MOVL    (R0), R5                                       ; 2181
                                54      35  A5  D0 00013     MOVL    53(R5), REC_SIZE
                  56            59          54  C1 00017     ADDL3   REC_SIZE, BUF_START, NEXT_REC                  ; 2185
                                53      FF  A6  9E 0001B     MOVAB   -1(R6), R3                                     ; 2189
                                5A          53  D1 0001F     CMPL    R3, BUF_END
                                            0A  1F 00022     BLSSU   1$
                                53          55  D0 00024     MOVL    R5, R3                                         ; 2191
                                52      04  AE  D0 00027     MOVL    CTX, R2
                                          F76E  31 0002B     BRW     PDP_BUFFER_CHECK
        54      39  A5      0C  BE      08  AE  2C 0002E 1$:  MOVC5   LEN, @BUF, 57(R5), REC_SIZE, (BUF_START)       ; 2195
                                            69     00036
                                53          56  D0 00037     MOVL    NEXT_REC, R3                                   ; 2199
                                52      04  AE  D0 0003A     MOVL    CTX, R2
                                          F78E  31 0003E     BRW     PDP_BUFFER_UPDATE

; Routine Size:  65 bytes,    Routine Base:  EXCH$PDP_CODE + 0B67
```

```
1 GLOBAL ROUTINE pdp_put_stream (buf_start, buf_end, ctx : $ref_bblock, len, buf) : jsb_put =      %SBTTL 'pdp_
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Add the next stream record in the file
2 !
2 ! INPUTS:
2 !
2 !       buf_start - Pointer to next byte in the buffer
2 !       buf_end   - Pointer to one past the end of buffer
2 !       ctx       - Output file context block
2 !       len       - Length of the record to be put
2 !       buf       - Address of the record
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       see the BIND expression
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       see the BIND expression
2 !
2 ! ROUTINE VALUE:
2 !
2 !       true if success, false if any error
2 !
2 ! SIDE EFFECTS:
2 !
2 !       error conditions will be signaled
2 !--
2
2 $dbgtrc_prefix ('pdp_put_stream> ');
2
2 REGISTER
2     actual_len,
2     next_rec,
2     tmp
2     ;
2
2 BIND
2     copy = exch$a_gbl [excg$a_copy_work]: $ref_bblock,  ! COPY verb work area
2     out_filb = copy [copy$a_out_filb]   : $ref_bblock   ! pointer to filb for an open Files-11 output file
2     ;
2
2 $debug_print_fao ('entry, len=!UL, buf[0:19]="!AF"', .len, 20, .buf);
2
2 ! Get a pointer to the start of the next record after this one
2 !
2 next_rec = .buf_start + .len + 2;                 ! Assume record plus <CR><LF>
2
2 ! See if the next record will fit in the buffer, EOF or advance the buffer if it isn't
```

```
; 2186          2259  2 !
; 2187          2260  2 IF (.next_rec - 1) GEQU .buf_end
; 2188          2261  2 THEN
; 2189          2262  2     RETURN pdp_buffer_check (.ctx, .out_filb);
; 2190          2263  2
; 2191          2264  2 ! Move the record to the buffer
; 2192          2265  2 !
; 2193          2266  2 actual_len = pdp_copy_stream_record (.len, .buf, .buf_start);
; 2194          2267  2
; 2195          2268  2 ! Update the next record position and return
; 2196          2269  2 !
; 2197          2270  2 RETURN pdp_buffer_update (.ctx, .buf_start + .actual_len);
; 2198          2271  2
; 2199          2272  1 END;



                     54             59  D0 00000 PDP_PUT_STREAM::
                                                          MOVL    R9, R4                                  ; 2202
         50 00000000G EF            04  C1 00003          ADDL3   #4, EXCH$A_GBL, R0                      ; 2248
         51           60 00000044   8F  C1 0000B          ADDL3   #68, (R0), R1                           ; 2249
         59           54        08  AE  C1 00013          ADDL3   LEN, BUF_START, R9                      ; 2256
                      50        02  A9  9E 00018          MOVAB   2(R9), NEXT_REC                         ;
                                    50  D7 0001C          DECL    R0                                      ; 2260
                      5A            50  D1 0001E          CMPL    R0, BUF_END                             ;
                                    0A  1F 00021          BLSSU   1$                                      ;
                      53            61  D0 00023          MOVL    (R1), R3                                ; 2262
                      52        04  AE  D0 00026          MOVL    CTX, R2                                 ;
                                  F72E  31 0002A          BRW     PDP_BUFFER_CHECK                        ;
                                    54  DD 0002D 1$:      PUSHL   BUF_START                               ; 2266
                                10  AE  DD 0002F          PUSHL   BUF                                     ;
                                10  AE  DD 00032          PUSHL   LEN                                     ;
                F8BD  CF            03  FB 00035          CALLS   #3, PDP_COPY_STREAM_RECORD              ;
         53           54            50  C1 0003A          ADDL3   ACTUAL_LEN, BUF_START, R3               ; 2270
                      52        04  AE  D0 0003E          MOVL    CTX, R2                                 ;
                                  F749  31 00042          BRW     PDP_BUFFER_UPDATE                       ;

; Routine Size:  69 bytes,    Routine Base:  EXCH$PDP_CODE + 0BAB
; 2201          2273  1 END
; 2202          2274  0 ELUDOM



                                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                       PSECT SUMMARY
;
;       Name                        Bytes                         Attributes
;
;  EXCH$PDP_CODE                        3053  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                               Library Statistics
;
;                                               --------- Symbols --------       Pages        Processing
;       File                                    Total    Loaded   Percent        Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                18619         3         0        1000          00:01.8
;  _$255$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1          1151        99         8          79          00:01.3




;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:EXCPDP/OBJ=OBJ$:EXCPDP MSRC$:EXCPDP/UPDATE=(ENH$:EXCPDP)

; Size:           3053 code + 0 data bytes
; Run Time:          00:57.4
; Elapsed Time:      02:38.6
; Lines/CPU Min:      2377
; Lexemes/CPU-Min: 21756
; Memory Used:  187 pages
; Compilation Complete
```

0162 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY